Dealer Event Catalog

An Event-Driven System Using Log-Based Architecture

[Prerequisites 2](#_Toc10534996)

[Installation 2](#_Toc10534997)

[Confluent Platform 2](#_Toc10534998)

[Elasticsearch 3](#_Toc10534999)

[Kibana 3](#_Toc10535000)

[Usage 3](#_Toc10535001)

[Kafka REST Proxy 3](#_Toc10535002)

[Producing JSON messages 3](#_Toc10535003)

[Consuming JSON Messages 3](#_Toc10535004)

[Inspecting Topics 4](#_Toc10535005)

[API Docs 4](#_Toc10535006)

[Additional Content Types 5](#_Toc10535007)

[Kafka 5](#_Toc10535008)

[Producing Messages 5](#_Toc10535009)

[Consuming Messages 5](#_Toc10535010)

[Environment Specific Changes 6](#_Toc10535011)

[Image Repos 6](#_Toc10535012)

[Security Context 6](#_Toc10535013)

[Chart Image Names 6](#_Toc10535014)

[Network policy 7](#_Toc10535015)

[Components 7](#_Toc10535016)

[Control Center 7](#_Toc10535017)

[Kafka 7](#_Toc10535018)

[Kafka Connect 7](#_Toc10535019)

[Kafka REST 8](#_Toc10535020)

[KSQL Server 8](#_Toc10535021)

[Schema Registry 8](#_Toc10535022)

[Zookeeper 8](#_Toc10535023)

[Additional Components 9](#_Toc10535024)

[Grafana 9](#_Toc10535025)

[Prometheus 9](#_Toc10535026)

[Prometheus Exporter for Kafka 9](#_Toc10535027)

[Postgres 9](#_Toc10535028)

[NiFi 9](#_Toc10535029)

[Elasticsearch 9](#_Toc10535030)

[Kibana 10](#_Toc10535031)

[Monitoring 10](#_Toc10535032)

[Prometheus 10](#_Toc10535033)

[Grafana 11](#_Toc10535034)

# Prerequisites

Kubernetes cluster

* Minikube for local development - [installation instructions](https://docs.confluent.io/current/installation/installing_cp/cp-helm-charts/docs/index.html#create-local-minikube)
* Or [ng-garage-nafta-qa.app.corpintra.net](http://ng-garage-nafta-qa.app.corpintra.net/) for non-local development

Helm installed on Kubernetes cluster - [installation instructions](https://helm.sh/docs/using_helm/#quickstart-guide).

**Important**

Helm must be initialized in a namespace where user has appropriate permissions, in this example namespace kafka is used:

helm init --tiller-namespace=kafka

# Installation

Execute the following commands to deploy Kafka, NiFi, Elasticsearch and related components.

## Confluent Platform

helm install helm-charts/confluent \

--name confluent \

--tiller-namespace=kafka \

--namespace=kafka \

--set \

cp-kafka-rest.external.enabled=true,\

cp-kafka-rest.external.type=ClusterIP,\

cp-kafka-rest.external.externalTrafficPolicy=null,\

cp-zookeeper.persistence.enabled=false,\

cp-kafka.persistence.enabled=false,\

cp-kafka-connect.persistence.enabled=false,\

cp-ksql-server.persistence.enabled=false,\

cp-schema-registry.persistence.enabled=false

## Elasticsearch

helm install helm-charts/elasticsearch \

--name elasticsearch \

--tiller-namespace=kafka \

--namespace=kafka

## Kibana

helm install helm-charts/kibana \

--name kibana \

--tiller-namespace=kafka \

--namespace=kafka

## NiFi

helm install helm-charts/nifi \

--name nifi \

--tiller-namespace=kafka \

--namespace=kafka

# Basic Usage

Once Helm charts have been successfully installed, Kafka can be accessed by clients external and internal to the Kubernetes cluster.

## Kafka REST Proxy

Due to limitations of current ingress and DNS, Kafka is unable to accept native connections from clients external to the Kubernetes cluster. Kafka REST Proxy provides an HTTP-based interface for those clients, as well as clients without native Kafka support.

### Producing JSON messages

1. Produce a message using JSON with the value '{ "foo": "bar" }' to jsontest topic

curl -X POST -L -k \

-H "Content-Type: application/vnd.kafka.json.v2+json" \

-H "Accept: application/vnd.kafka.v2+json" \

--data '{"records":[{"value":{"foo":"bar"}}]}' \

"https://ng-garage-nafta-qa.app.corpintra.net/topics/jsontest"

Expected response:

{"offsets":[{"partition":0,"offset":0,"error\_code":null,"error":null}],"key\_schema\_id":null,"value\_schema\_id":null}

### Consuming JSON Messages

1. Create a consumer for JSON data, starting at the beginning of the topic's log

curl -X POST \

-H "Content-Type: application/vnd.kafka.v2+json" \

--data '{"name": "my\_consumer\_instance", "format": "json", "auto.offset.reset": "earliest"}' \

https://ng-garage-nafta-qa.app.corpintra.net/consumers/my\_json\_consumer

{"instance\_id":"my\_consumer\_instance", "base\_uri":"http://localhost:8082/consumers/my\_json\_consumer/instances/my\_consumer\_instance"}

2. Subscribe to jsontest topic

curl -X POST \

-H "Content-Type: application/vnd.kafka.v2+json" \

--data '{"topics":["jsontest"]}' \

https://ng-garage-nafta-qa.app.corpintra.net/consumers/my\_json\_consumer/instances/my\_consumer\_instance/subscription

<empty response>

3. Consume data using the base URL in the first response

curl -X GET \

-H "Accept: application/vnd.kafka.json.v2+json" \ https://ng-garage-nafta-qa.app.corpintra.net/consumers/my\_json\_consumer/instances/my\_consumer\_instance/records

[{"key":null,"value":{"foo":"bar"},"partition":0,"offset":0,"topic":"jsontest"}]

4. Close the consumer with a DELETE to make it leave the group and clean up its resources

curl -X DELETE \

-H "Content-Type: application/vnd.kafka.v2+json" \

https://ng-garage-nafta-qa.app.corpintra.net/consumers/my\_json\_consumer/instances/my\_consumer\_instance

<empty response>

### Inspecting Topics

1. Get a list of topics

curl "https://ng-garage-nafta-qa.app.corpintra.net/topics"

["\_\_consumer\_offsets","\_schemas","avrotest","binarytest","jsontest"]

2. Get info about one topic

curl "https://ng-garage-nafta-qa.app.corpintra.net/topics/jsontest"

{"name":"avrotest","configs":{},"partitions":[{"partition":0,"leader":0,"replicas":[{"broker":0,"leader":true,"in\_sync":true}]}]}

3. Get info about a topic's partitions

curl "https://ng-garage-nafta-qa.app.corpintra.net/topics/jsontest/partitions"

[{"partition":0,"leader":0,"replicas":[{"broker":0,"leader":true,"in\_sync":true}]}]

### API Docs

Kafka REST Proxy exposes Kafka’s functionality via a REST API, for the full list of available commands visit <https://docs.confluent.io/current/kafka-rest/api.html>

### Additional Content Types

In addition to JSON messages, Kafka REST Proxy supports Avro and Binary formats. For more information visit:

Content Types Overview – <https://docs.confluent.io/current/kafka-rest/api.html#content-types>

Avro Messages - <https://docs.confluent.io/current/kafka-rest/quickstart.html#produce-and-consume-avro-messages>

Binary Messages - <https://docs.confluent.io/current/kafka-rest/quickstart.html#produce-and-consume-binary-messages>

## Kafka

### Producing Messages

To connect from a client pod:

1. Deploy a kafka client pod with configuration:

kubectl apply -f clients/kafka-client.yaml -n kafka

2. Log into the Pod

kubectl exec -it kafka-client -n kafka -- /bin/bash

### Creating a Topic

kafka-topics --zookeeper confluent-cp-zookeeper-headless:2181 --topic confluent-topic --create --partitions 1 --replication-factor 1 --if-not-exists

### Producing Messages

MESSAGE="`date -u`"

echo "$MESSAGE" | kafka-console-producer --broker-list confluent-cp-kafka-headless:9092 --topic confluent-topic

### Consuming Messages

kafka-console-consumer --bootstrap-server confluent-cp-kafka-headless:9092 --topic confluent-topic --from-beginning --timeout-ms 2000 --max-messages 1 | grep "$MESSAGE"

# Dealer Event Catalog

## Setup

### Elasticsearch

Create a mapping for the timestamp field, letting Elasticsearch know it is of type date.

kubectl exec elasticsearch-0 -n kafka -- curl -XPUT \ "localhost:9200/messages?pretty" \

-H 'Content-Type: application/json' \

-d '{"mappings": {"message": {"properties": {"timestamp": { "type": "date" }}}}}'

### Kibana

Forward Kibana to local port 5601.

kubectl port-forward -n kafka kibana-0 5601: 5601

### NiFI

Forward NiFi to local port 8080.

kubectl port-forward -n kafka nifi-0 8080:8080

Create raw event viewer flow.

response=$(curl -F template=@event-catalog/flow-dealer-updates.xml http://localhost:8080/nifi-api/process-groups/root/templates/upload -L -k)

template=$(echo ${response}|sed -n 's:.\*<id>\(.\*\)</id>.\*:\1:p')

data="{ \"originX\": 0.0, \"originY\": 0.0, \"templateId\": \"${template}\" }"

curl http://localhost:8080/nifi-api/process-groups/root/template-instance \

-H "Content-Type: application/json" \

-d "${data}"

### Kafka Connect

kubectl port-forward svc/confluent-cp-kafka-connect 8083:8083 -n kafka

curl http://localhost:8083/connectors \

-H "Content-Type: application/json" \

-d@event-catalog/connector-dealer-db.json

# Environment Specific Changes

This section outlines modifications made to official Docker images and Helm charts in order to leverage them in the ng-garage-nafta-qa environment. Modified source code for images and charts is included in this project.

## Image Repos

All images had to be pulled from official repos, re-tagged and pushed to reg-dhc-americas.app.corpintra.net prior to being usable in Kubernetes deployments. This example demonstrates the process for confluentinc/cp-zookeeper:5.2.1 image.

# pull the image

docker pull confluentinc/cp-zookeeper:5.2.1

# tag image for internal repo

docker tag confluentinc/cp-zookeeper:5.2.1 reg-dhc-americas.app.corpintra.net/myanama/cp-zookeeper:5.2.1

# push image to internal repo

docker push reg-dhc-americas.app.corpintra.net/myanama/cp-zookeeper:5.2.1

## Security Context

Majority of charts were modified to include a security context section in order to work around ng-garage’s Kubernetes cluster policy of not using root users. For example:

...

spec:

**securityContext:**

**fsGroup: 1000**

**runAsUser: 1000**

containers:

- name: prometheus-jmx-exporter

...

## Chart Image Names

All charts values.yaml files were updated to use images from reg-dhc-americas.app.corpintra.net repo and with original image tags. For example:

...

cp-zookeeper:

enabled: true

servers: 3

**image: reg-dhc-americas.app.corpintra.net/myanama/cp-zookeeper**

imageTag: 5.2.1

...

## Network policy

By default, components are not able to communicate with other components due to ng-garage-nafta-qa‘s default network policy. With the exception of Postgres, NetworkPolicy‘s had to be created for each component, here’s a sample one for Kafka service.

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: **cp-kafka**-network-policy

spec:

podSelector:

matchLabels:

app: **cp-kafka**

ingress:

- {}

# Components

### Control Center

|  |  |
| --- | --- |
| Chart | cp-helm-charts/tree/master/charts/cp-control-center |
| Original image | confluentinc/cp-enterprise-control-center:5.2.0 |
| New image | myanama/cp-enterprise-control-center:5.2.0 |

### Kafka

|  |  |
| --- | --- |
| Chart | cp-helm-charts/tree/master/charts/cp-kafka |
| Original image | confluentinc/cp-enterprise-kafka:5.2.1 |
| New image | myanama/cp-enterprise-kafka:5.2.1 |
| Documentation |  |

### Kafka Connect

|  |  |
| --- | --- |
| Chart | cp-helm-charts/tree/master/charts/cp-kafka-connect |
| Original image | confluentinc/cp-enterprise-kafka-connect:5.2.1 |
| New image | myanama/cp-enterprise-kafka-connect:5.2.1 |
| Image changes | * Installed Debezium Postgres connector |

### Kafka REST

|  |  |
| --- | --- |
| Chart | cp-helm-charts/tree/master/charts/cp-kafka- rest |
| Original image | confluentinc/cp-enterprise-kafka- rest:5.2.1 |
| New image | myanama/cp-enterprise-kafka- rest:5.2.1 |
| Documentation | <https://docs.confluent.io/current/kafka-rest/quickstart.html> |

### KSQL Server

|  |  |
| --- | --- |
| Chart | cp-helm-charts/tree/master/charts/cp-ksql-server |
| Original image | confluentinc/cp-ksql-server:5.2.1 |
| New image | myanama/cp-ksql-server:5.2.1 |
| Image changes | * Created kafka group * Created kafka user * Granted ownership of KSQL Server directory to kafka user |

### Schema Registry

Chart: cp-helm-charts/tree/master/charts/cp-schema-registry

Original image: confluentinc/cp-schema-registry:5.2.1

New image: myanama/cp-schema-registry:5.2.1

### Zookeeper

|  |  |
| --- | --- |
| Chart | cp-helm-charts/tree/master/charts/cp-zookeeper |
| Original image | confluentinc/cp-zookeeper:5.2.1 |
| New image | myanama/cp-zookeeper:5.2.1 |
| Connection string for Confluent Kafka:    confluent-cp-zookeeper-0.confluent-cp-zookeeper-headless:2181,confluent-cp-zookeeper-1.confluent-cp-zookeeper-headless:2181,...  To connect from a client pod:  1. Deploy a zookeeper client pod with configuration:  kubectl apply -f clients/zookeeper-client.yaml  2. Log into the Pod  kubectl exec -it zookeeper-client -- /bin/bash  3. Use zookeeper-shell to connect in the zookeeper-client Pod:  zookeeper-shell confluent-cp-zookeeper:2181  4. Explore with zookeeper commands, for example:  # Gives the list of active brokers  ls /brokers/ids  # Gives the list of topics  ls /brokers/topics  # Gives more detailed information of the broker id '0'  get /brokers/ids/0 | |

## Additional Components

### Grafana

Original image: grafana/grafana:6.0.2

New image: myanama/grafana:6.0.2

### Prometheus

Original image: prom/prometheus:v2.8.0

New image: myanama/prometheus:v2.8.0

### Prometheus Exporter for Kafka

Original image: solsson/kafka-prometheus-jmx-exporter@sha256:6f82e2b0464f50da8104acd7363fb9b995001ddff77d248379f8788e78946143

New image: myanama/prometheus-exporter:0.3.1

### Postgres

Original image: debezium/postgres:10

New image: myanama/postgres:10

### NiFi

Original image: apache/nifi:1.8.0

New image: myanama/nifi:1.8.0

### Elasticsearch

|  |  |
| --- | --- |
| Chart | Charts/stable/elasticsearch |
| Original image | docker.elastic.co/elasticsearch /elasticsearch -oss:6.6.1 |
| New image | myanama/elasticsearch -oss:6.6.1 |
| Elasticsearch can be accessed:  \* Within your cluster, at the following DNS name at port 9200:  elasticsearch-client.default.svc  \* From outside the cluster, run these commands in the same shell:  export POD\_NAME=$(kubectl get pods --namespace default -l "app=elasticsearch,component=client,release=elasticsearch" -o jsonpath="{.items[0].metadata.name}")  echo "Visit http://127.0.0.1:9200 to use Elasticsearch"  kubectl port-forward --namespace default $POD\_NAME 9200:9200 | |

Original image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.6.1

New image: myanama/elasticsearch-oss:6.6.1

### Kibana

|  |  |
| --- | --- |
| Chart | Charts/stable/kibana |
| Original image | docker.elastic.co/kibana/kibana-oss:6.6.1 |
| New image | myanama/kibana-oss:6.6.1 |
| Kibana can be accessed:  \* From outside the cluster, run these commands in the same shell:  export POD\_NAME=$(kubectl get pods --namespace default -l "app=kibana,release=kibana" -o jsonpath="{.items[0].metadata.name}")  echo "Visit http://127.0.0.1:5601 to use Kibana"  kubectl port-forward --namespace default $POD\_NAME 5601:5601 | |

## Monitoring

### Prometheus

helm install stable/prometheus \

--name prometheus \

--tiller-namespace=garagetest \

--namespace=garagetest \

--set \

rbac.create="false",\

alertmanager.image.repository="reg-dhc-americas.app.corpintra.net/myanama/alertmanager",\

alertmanager.image.tag="0.15.3",\

configmapReload.image.repository="reg-dhc-americas.app.corpintra.net/myanama/configmap-reload",\

initChownData.image.repository="reg-dhc-americas.app.corpintra.net/myanama/busybox",\

kubeStateMetrics.image.repository="reg-dhc-americas.app.corpintra.net/myanama/kube-state-metrics",\

nodeExporter.image.repository="reg-dhc-americas.app.corpintra.net/myanama/node-exporter",

server.image.repository="reg-dhc-americas.app.corpintra.net/myanama/prometheus",

pushgateway.image.repository="reg-dhc-americas.app.corpintra.net/myanama/pushgateway",

initChownData.enabled="false",

server.persistentVolume.enabled="false",

alertmanager.persistentVolume.enabled="false",

pushgateway.persistentVolume.enabled="false"

### Grafana

helm install stable/grafana \

--name grafana \

--tiller-namespace=garagetest \

--namespace=garagetest \

--set \

rbac.create="false",\

rbac.pspEnabled="false",\

rbac.pspUseAppArmor="false",\

image.repository="reg-dhc-americas.app.corpintra.net/myanama/grafana",\

downloadDashboardsImage.repository="reg-dhc-americas.app.corpintra.net/myanama/curl",\

initChownData.enabled="false"

* Charts
* nodeExporter.image.repository="reg-dhc-americas.app.corpintra.net/myanama/node-exporter",
* server.image.repository="reg-dhc-americas.app.corpintra.net/myanama/prometheus",
* pushgateway.image.repository="reg-dhc-americas.app.corpintra.net/myanama/pushgateway",
  + Ingress
* Basic Commands
  + Kafka
    - Native
    - REST
  + Zookeeper
* Application
  + NiFi flows
  + Postgres schema and data
  + Elasticsearch schema
* Future tasks
  + Operations
    - Logging
    - Monitoring
    - Alerting
  + Clustered/HA
    - NiFi
    - Elasticsearch
    - Kibana
  + Persistence
  + Encryption
    - In-flight
    - At-rest
  + Role based access control
    - Authentication
    - Authorization
  + Ingress
    - TCP
    - Subdomain-based FQDNs
  + Testing
    - Message size
    - Message volume
    - Producer count
    - Consumer count
    - Endurance
  + Licenses
    - Confluent
    - Elastic
* Aspects
  + **Collection**
  + **Storage**
  + **Processing**
  + **Visualization**
  + **Access**