Configurazione di un cluster Kafka con un singolo nodo su Kubernetes

Dopo aver configurato un cluster di Kubernetes di un singolo nodo, essersi connessi al cluster in modo che i comandi dati tramite il tool kubectl effettivaemnte modifichino lo stato del cluster di cui sopra e aver inizializzato Helm;
eseguire i seguenti comandi:

```
# If not already done
helm init

# Add incubator charts repository
helm repo add incubator https://kubernetes-charts-incubator.storage.googleapis.com/

# Create kafka namespace and assign as default namespace for subsequent commands
kubectl create ns kafka
kubectl config set-context $(kubectl config current-context) --namespace kafka

# Install a single node zookeeper + kafka cluster
helm install --name raiden-zookeeper --set replicaCount=1 incubator/zookeeper
helm install --name raiden-kafka -f custom-kafka-conf.yaml incubator/kafka --version 0.13.1
```

Dove il contenuto del file `custom-kafka-conf.yaml` è il seguente:

```
replicas: 1
configurationOverrides:
  "offsets.topic.replication.factor": 1
  "auto.leader.rebalance.enable": true
  "auto.create.topics.enable": true
  "confluent.support.metrics.enable": false
  "advertised.listeners": |-
    EXTERNAL://raiden-kafka-${KAFKA_BROKER_ID}-external.kafka.svc.cluster.local:$((31090 + ${K
  "listener.security.protocol.map": |-
    PLAINTEXT:PLAINTEXT,EXTERNAL:PLAINTEXT
  "listeners": "PLAINTEXT://0.0.0.0:9092,EXTERNAL://0.0.0.0:19092"
external:
  enabled: true
  domain: "kafka.svc.cluster.local"
topics:
  - name: raiden-events
    config: "compression.type=snappy,retention.ms=-1"
zookeeper:
  # Since we installed zookeeper ourselves we do not need to deploy it again
  enabled: false

  # For the same reason we now need to provide a zookeeper endpoint
  url: "raiden-zookeeper-headless.kafka.svc.cluster.local"
  port: 2181
```

Ogni produttore può connettersi al cluster tramite l'indirizzo del servizio *headless* **raiden-kafka-headless** alla porta *9092*.

```
p =  Producer({"bootstrap.servers": "raiden-kafka-headless:9092"})
```

Ogni produttore potrà connettersi al cluster tramite l'indirizzo del servizio **raiden-kafka** alla porta *9092*.

```
c =  Consumer({"bootstrap.servers": "raiden-kafka:9092", "group.id": "cacher"})
```

N.B.: I broker comunicheranno tra loro attraverso gli indirizzi contenuti nel parametro `advertised.listeners` di ogni broker. Come da configurazione, il broker di id `1` accetterà comunicazioni su `EXTERNAL://raiden-kafka-1-external.kafka.svc.cluster.local:31091`. Controllare questo link per chiarire eventuali dubbi.

Confluent Schema Registry per utilizzare Apache Avro per la serializzazione

Per utilizzare produttori e consumatori Avro è necessario schierare anche un'istanza del Confluent Schema Registry.

Per installarlo è necessario eseguire questo comando:

```
helm install --name raiden-sr -f custom-schema-conf.yaml incubator/schema-registry
```

Dove il contenuto di `custom-schema-conf.yaml` è il seguente:

```
imageTag: 5.0.1
replicaCount: 1
servicePort: 8081
kafkaStore:
  overrideBootstrapServers: "PLAINTEXT://raiden-kafka-headless:9092"
  overrideGroupId: "raiden-sr-group"
sasl:
  scram:
    enabled: false
kafka:
  enabled: false
ingress:
  enabled: false
```

Produttori e consumatori all'interno del cluster potranno connettersi al registro tramite `HTTP` a questo indirizzo `http://raiden-sr-schema-registry:8081`

Produttori e Consumatori

A meno che non si stiano utilizzando le API Java ufficiali del progetto Apache, altri client di cui possiamo fidarci sono quelli sviluppati e mantenuti da Confluent.
Riguardo questi ultimi, sono tutti (Java a parte) basati su libdrkafka. Una lista esclusiva dei parametri di configurazione può essere trovata qui.

Per sviluppare in Python è possibile partire da questa immagine (usare almeno il tag 0.1.1 se si intende utilizzare Avro per serializzare e deserializzare).

Fornire accesso dall'esterno

Per permettere a processi esterni al cluster di comunicare con le risorse interne è necessario installare un ingress controller.
Utilizzeremo un nginx-ingress controller.

Per configurare un controller che consenta l'accesso ai nostri servizi è necessario eseguire questo comando:

```
helm install --name raiden-ingress -f custom-ingress-conf.yaml stable/nginx-ingress
```

Dove il contenuto del file `custom-ingress-conf.yaml` è il seguente:

```
controller:
  service:
    # This is must be a public IP owned by you.
    loadBalancerIP: "51.144.48.112"

rbac:
  create: false

# This will create a ConfigMap to consent external access to our
# kafka cluster and schema registry through TCP
tcp: {
  9092: "kafka/raiden-kafka-headless:9092",
  8081: "kafka/raiden-sr-schema-registry:8081"
}
```

> Per fornire correttamente l'accesso ai broker di kafka è necessario che venga dato a loro stessi un ingress e che essi stessi effettuino l'advertising di tale endpoint, altrimenti dopo aver contattato il servizio kafka-headess i broker comunicheranno il loro indirizzo IP *interno* al cluster k8s.