IBM Event Streams
Performance Report

25 October 2022

For Event Streams 11.0.4 running on Red Hat OpenShift Container Platform 4.10.32

Table of Contents

Disclaimer

Warranty and Liability Exclusion

Errors and Omissions

Local Availability

Alternative Products and Services

Trademarks and Service Marks

Export Regulations

Overview

This report contains indicative numbers that demonstrate the ability of IBM Event Streams to handle different levels of messaging traffic. The values were achieved using example workloads. It is not a definitive guide to peak performance capabilities but aims to show what Event Streams can handle based on examples. Performance will always depend on numerous factors including message throughput, message size, hardware, configuration settings, and so on.

Testing was based on Event Streams version 11.0.4 running on Red Hat OpenShift Container Platform 4.10.32 which was provisioned using Red Hat OpenShift on IBM Cloud.

This report will be updated on a regular basis. This is the third version of the report. The previous version was produced for Event Streams version 2019.4.1 on IBM Cloud Private 3.2.1.

Workloads

The table below illustrates the workloads measured during the creation of this report.

Workload	**Description**
Resilient	Demonstrates scaling as cluster size increases, favoring resilience over throughput, 3-way replication, all acknowledgements
Fast	Demonstrates scaling as cluster size increases, favoring throughput over resilience, no replication, leader acknowledgements
Payload size	Demonstrates the effect of message size on throughput

The workloads were measured without the Kafka proxy and with the Kafka proxy to demonstrate the differences when collecting producer metrics and without. Each set of figures and graphs show the values without the proxy first and with the proxy second.

Workload – Resilient

This workload measures throughput at increasing cluster sizes favoring resilience over throughput:

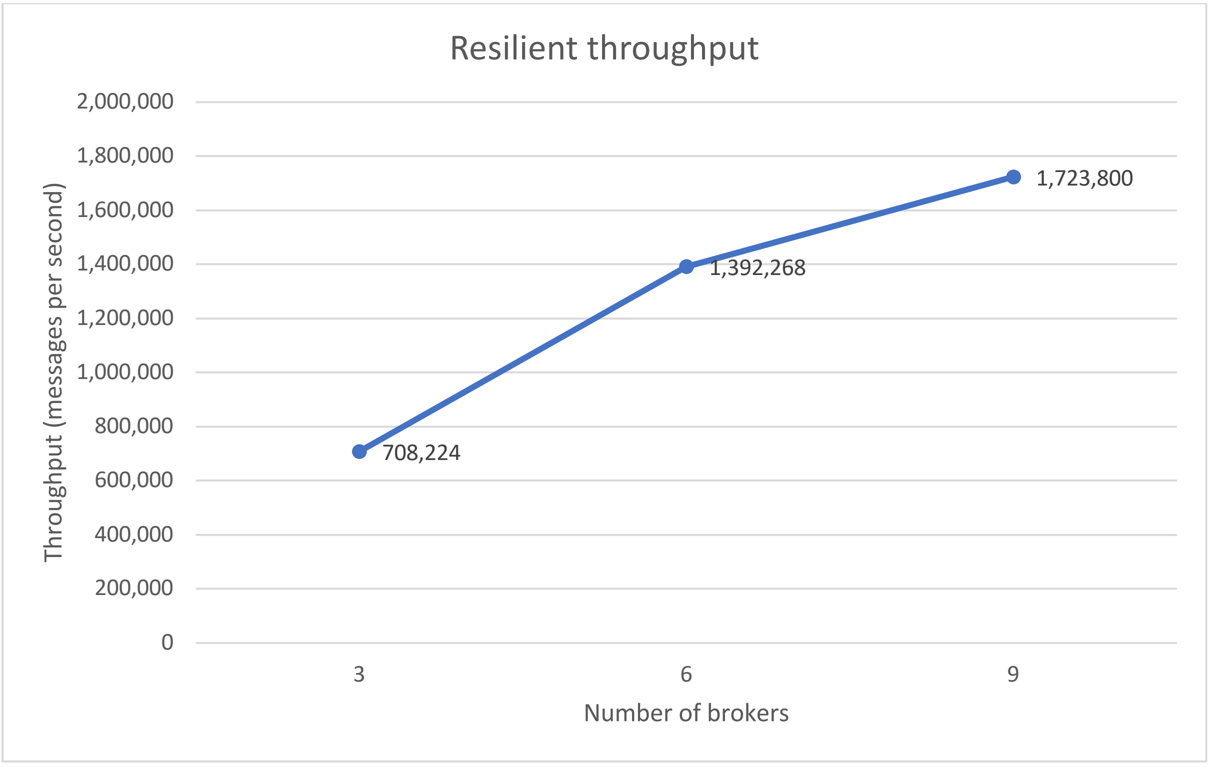
- Replication and acknowledgements protect messages against failures
- Messages are replicated across 3 brokers (replication factor 3)
- Acknowledgements from all brokers (acks=all)

A single topic is used for the messages. The number of partitions is double the number of brokers and the partition leadership and replicas are spread evenly across the brokers. The message payload is 128 bytes long. All consumers are in the same consumer group and each message is consumed by one consumer. The workload is generated by the Apache Kafka performance tests supplied within a standard Kafka installation and invoked by the `kafkaproducer-perf-test.sh` and `kafka-consumer-perf-test.sh` scripts. The throughput numbers are collected from the Kafka producer instances and aggregated to produce the overall message throughput.

The proxy numbers do not include a 9 broker sample because the test highlighted environmental impacts causing bottlenecks that skewed the numbers.

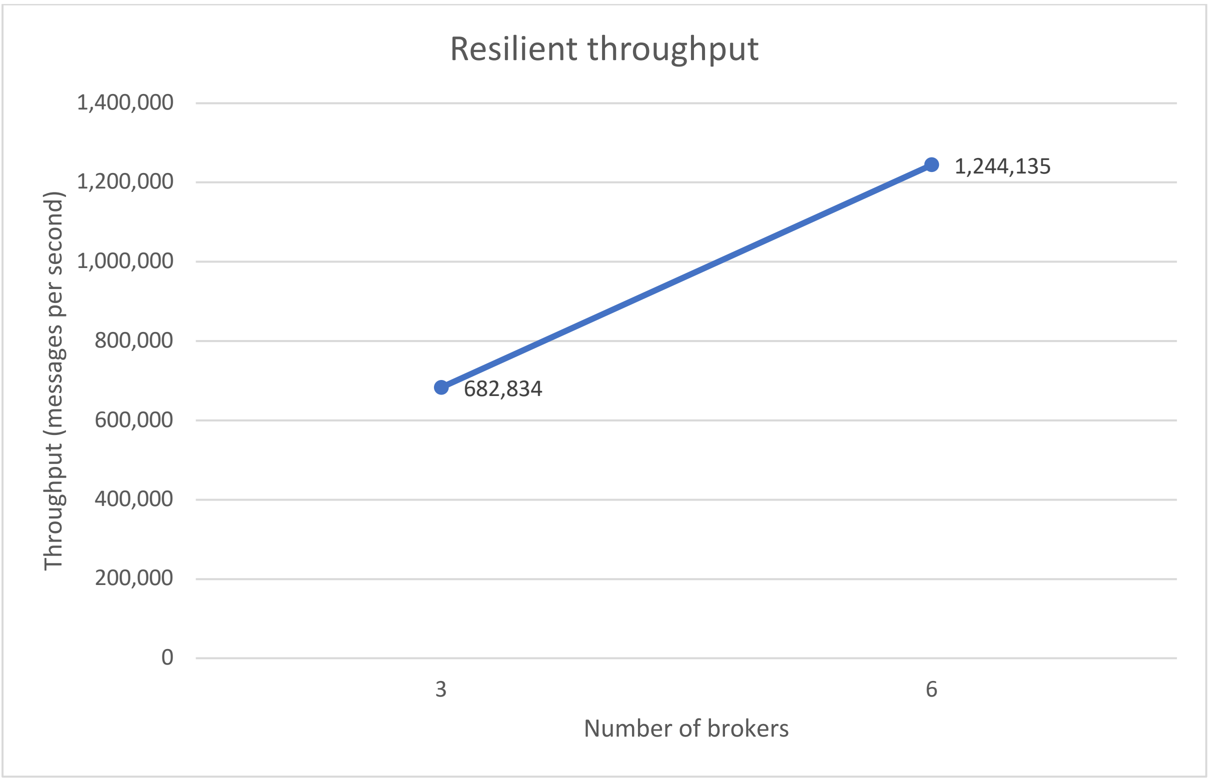
Performance without the proxy:

Brokers	**Consumers**	**Producers**	**Partitions**	**Throughput (messages/second)**
3	6	6	6	708,224
6	12	12	12	1,392,268
9	18	18	18	1,723,800



Performance with the proxy:

Brokers	Consumers	Producers	Partitions	Throughput (messages/second)
3	6	6	6	682,834
6	12	12	12	1,244,135



Workload – Fast

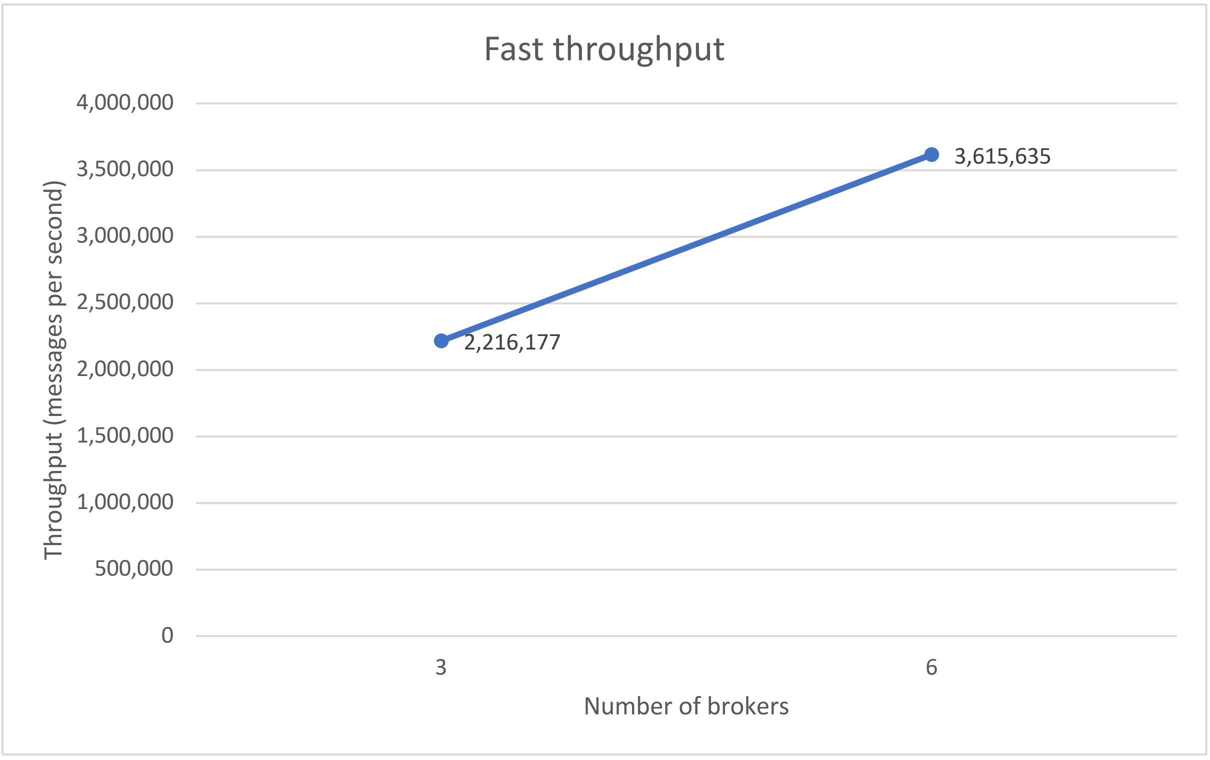
This workload measures throughput at increasing cluster sizes, favoring throughput over resilience:

- Lower overhead gives additional throughput but less resilience to failures
- The messages are not replicated across brokers (replication factor 1)
- Acknowledgements from 1 broker (acks=1)

A single topic is used for the messages. The number of producers and consumers are balanced to achieve constant message throughput. The number of partitions is set to match the number of consumers to evenly distribute message processing across the consumer group. The message payload is 128 bytes long. All consumers are in the same consumer group and each message is consumed by one consumer. The workload is generated by the Apache Kafka performance tests supplied within a standard Kafka installation and invoked by the `kafka-producer-perf-test.sh` and `kafka-consumer-perf-test.sh` scripts. The throughput numbers are collected from the Kafka producer instances and aggregated to produce the overall message throughput.

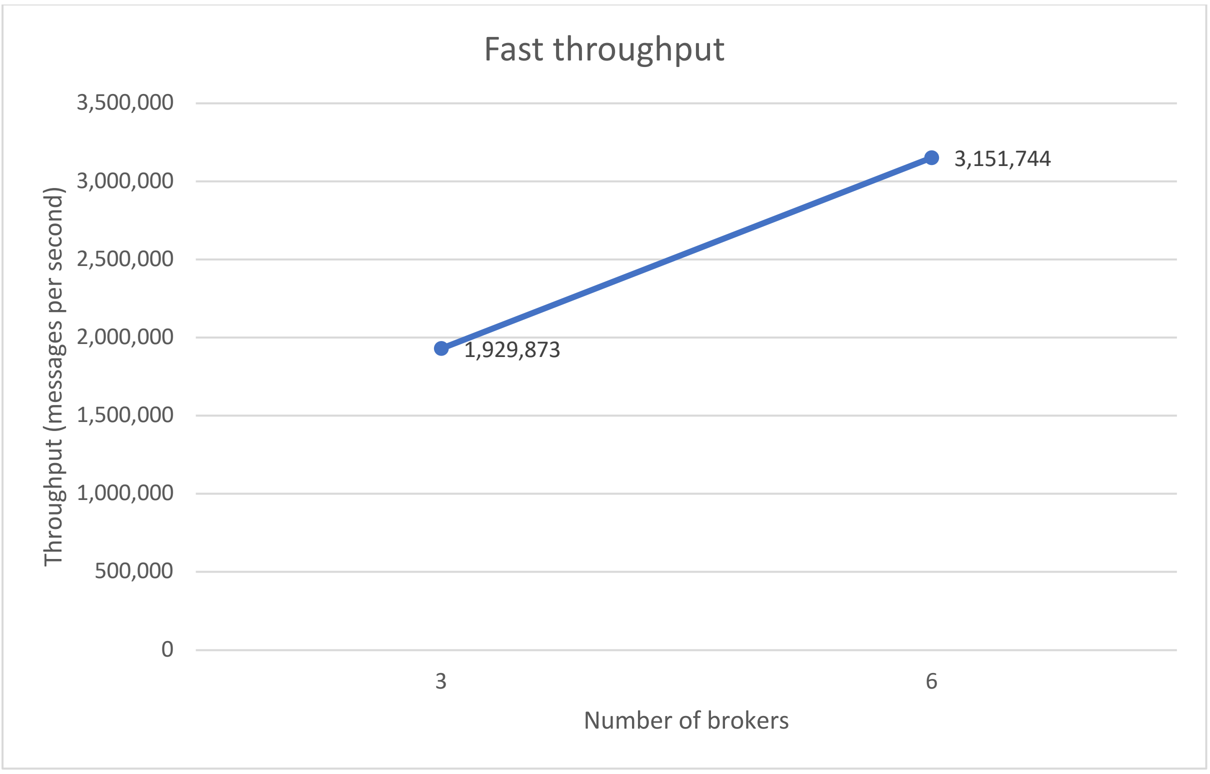
Performance without the proxy:

Brokers	Producers	Consumers	Partitions	Throughput (messages/second)
3	12	24	24	2,216,177
6	24	48	48	3,615,635



Performance with the proxy:

Brokers	Producers	Consumers	Partitions	Throughput (messages/second)
3	12	24	24	1,929,873
6	24	48	48	3,151,744

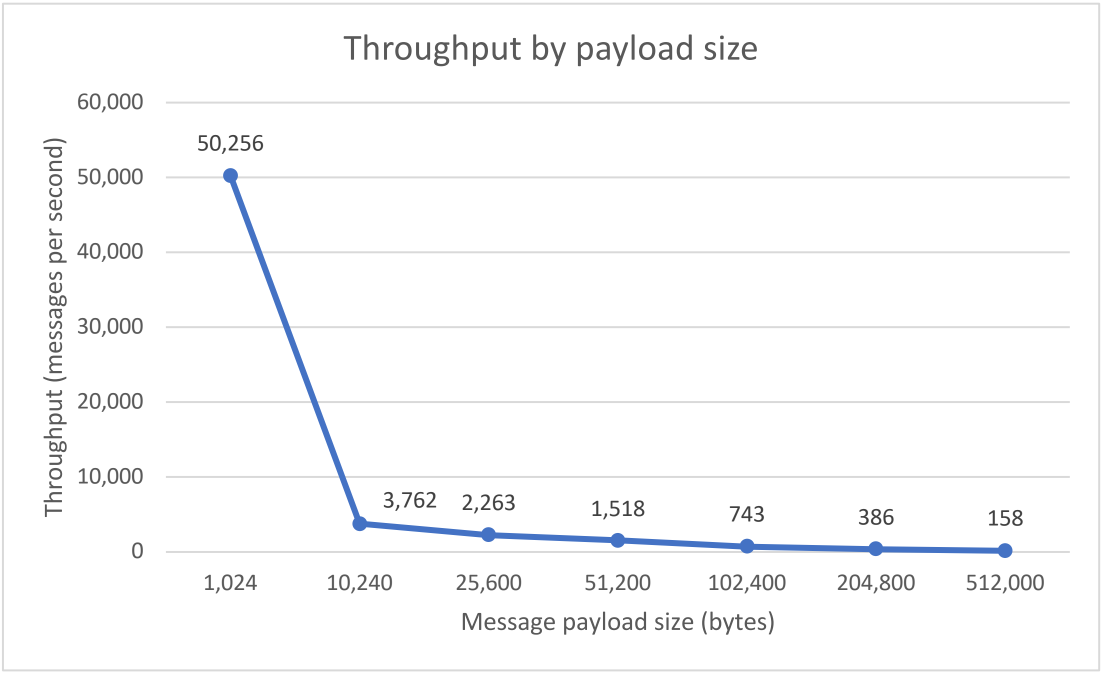


Workload – Payload Size

This test case measures the effect of increasing payload size on throughput. A single topic with one partition and no replication is used for the messages. One producer and no consumers are used. The workload is generated by the Apache Kafka performance tests supplied within a standard Kafka installation and invoked by the `kafka-producer-perf-test.sh` and `kafka-consumer-perf-test.sh` scripts. The throughput numbers are collected from the Kafka producer instances and aggregated to produce the overall message throughput.

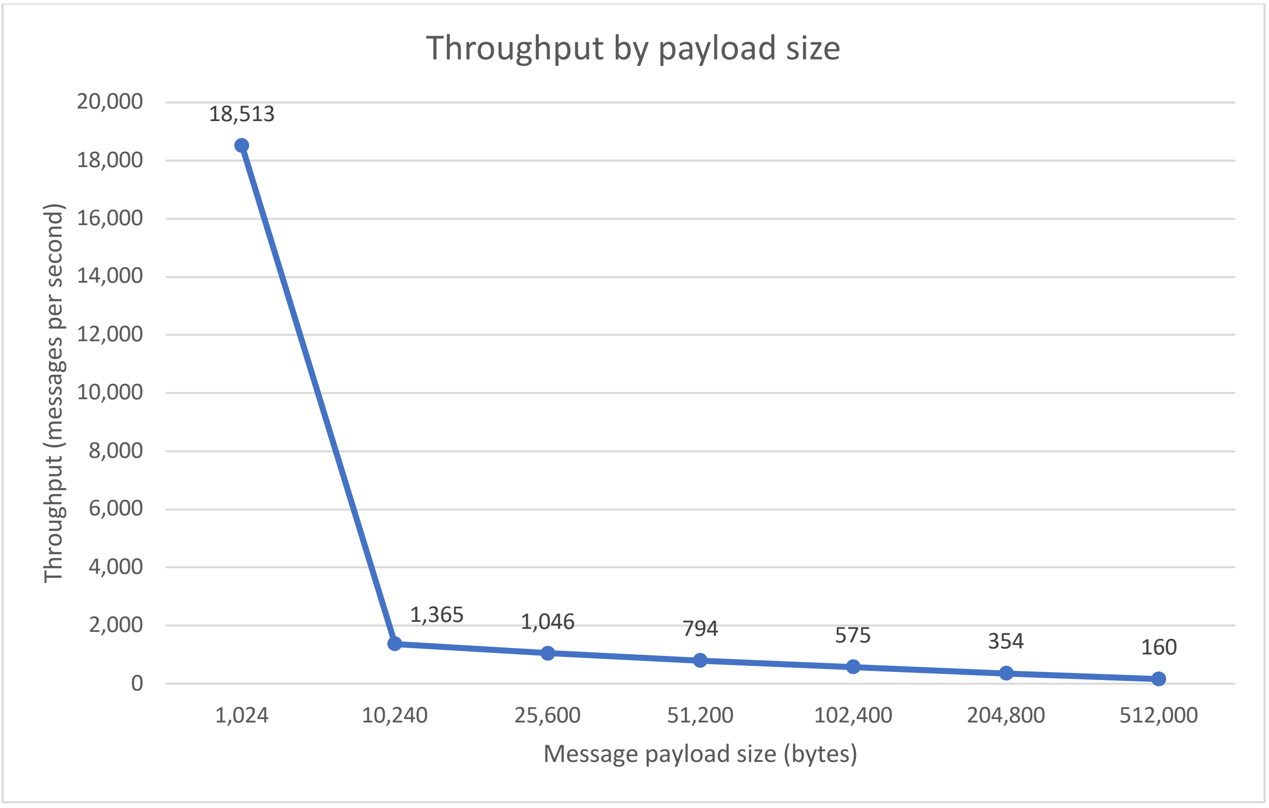
Performance without the proxy:

Message size (bytes)	Throughput (messages/second)	Throughput (MB/sec)
1,024	50,256	49.08
10,240	3,762	36.74
25,600	2,263	55.27
51,200	1,518	74.13
102,400	743	72.57
204,800	386	75.48
512,000	156	76.65



Performance with the proxy:

Message size (bytes)	Throughput (messages/second)	Throughput (MB/sec)
1,024	18,513	18.08
10,240	1,365	13.33
25,600	1,046	25.54
51,200	794	38.80
102,400	575	56.20
204,800	354	69.31
512,000	160	78.39



Test Configurations

The Kafka server was run on a Red Hat OpenShift on IBM Cloud cluster. The version of Red Hat OpenShift Container Platform was 4.10.32 The cluster was provisioned with 8 nodes with all 8 nodes being designated as both master and worker nodes. Each node was provisioned with 16 CPU and 128GB memory. IBM Cloud storage was preferred over OpenShift Data Foundation as the standard configuration for block storage had a better bandwidth capability. The persistent storage used the 10-iops non-retained storage class.

The producers and consumers were run on a separate cluster. The cluster was running Red Hat OpenShift Container Platform on IBM Cloud and was provisioned with 3 nodes running 8 CPU and 32GB memory. The clients were distributed evenly across the cluster using `Kubernetes topologySpreadConstraints`. The clients were run in parallel pods with each pod running `the kafka-producer-perf-test.sh` or `kafka-consumer-perf-test.sh` scripts which are shipped with Kafka. The clients used Kafka 3.2.1.

There was no additional tuning done on the client or server side and the standard template for a Production 3 brokers custom resource that is supplied with Event Streams was used. The only modification to the custom resources was the license accept and specifying the storage classes.

The Event Streams custom resource and the producer and consumer configuration are below.

Event Streams Custom Resource

The following custom resource configuration was used on all tests. The only change between tests were the number of replicas (brokers) in the Kafka section:

```yaml
apiVersion: eventstreams.ibm.com/v1beta2
kind: EventStreams
metadata:
  name: prod-3-brokers
spec:
  license:
    accept: true
    use: CloudPakForIntegrationProduction
  requestIbmServices:
    iam: true
    monitoring: true
  strimziOverrides:
    kafka:
      authorization:
        authorizerClass: com.ibm.eventstreams.runas.authorizer.RunAsAuthorizer
        supportsAdminApi: true
        type: custom
      config:
        num.network.threads: 9
        inter.broker.protocol.version: '3.2'
        log.cleaner.threads: 6
        num.io.threads: 24
        num.replica.fetchers: 3
        min.insync.replicas: 2
        log.message.format.version: '3.2'
        offsets.topic.replication.factor: 3
        default.replication.factor: 3
      listeners:
        - authentication:
            type: scram-sha-512
          name: external
          port: 9094
          tls: true
          type: route
        - authentication:
            type: tls
          name: tls
          port: 9093
          tls: true
          type: internal
      metricsConfig:
        type: jmxPrometheusExporter
        valueFrom:
          configMapKeyRef:
            key: kafka-metrics-config.yaml
```

```yaml
              name: prod-3-brokers-metrics-config
        replicas: 3
        resources:
          limits:
            cpu: 4000m
            memory: 8096Mi
          requests:
            cpu: 4000m
            memory: 8096Mi
        storage:
          class: ibmc-vpc-block-10iops-tier
          size: 100Gi
          type: persistent-claim
      zookeeper:
        metricsConfig:
          type: jmxPrometheusExporter
          valueFrom:
            configMapKeyRef:
              key: zookeeper-metrics-config.yaml
              name: prod-3-brokers-metrics-config
        replicas: 3
        storage:
          class: ibmc-vpc-block-10iops-tier
          size: 4Gi
          type: persistent-claim
    adminUI: {}
    restProducer: {}
    apicurioRegistry: {}
    adminApi: {}
    collector: {}
    version: 11.0.4
```

Producer Properties

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: producer-properties
  namespace: es
data:
  producer.properties: |-
    ############################# Producer Basics #############################

    # list of brokers used for bootstrapping knowledge about the rest of the
cluster
    bootstrap.servers=prod-3-brokers-kafka-bootstrap-es-ibm.itzroks-06000081s7-
o0sfdl-6ccd7f378ae819553d37d5f2ee142bd6-0000.eu-gb.containers.appdomain.cloud:443

    # specify the compression codec for all data generated: none, gzip, snappy,
lz4, zstd
    compression.type=none

    # reliability
    acks=all

    # SCRAM Properties
    security.protocol=SASL_SSL
    sasl.mechanism=SCRAM-SHA-512
    sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule
required username="<username>" password="<password>";
    # TLS Properties
    ssl.protocol=TLSv1.2
    ssl.truststore.location=/certs/cluster/ca.p12
    ssl.truststore.password=<truststore-password>
    ssl.truststore.type=PKCS12
```

Consumer Properties

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: consumer-properties
  namespace: es
data:
  consumer.properties: |-
    # list of brokers used for bootstrapping knowledge about the rest of the
cluster
    bootstrap.servers=prod-3-brokers-kafka-bootstrap-es-ibm.itzroks-06000081s7-
o0sfdl-6ccd7f378ae819553d37d5f2ee142bd6-0000.eu-gb.containers.appdomain.cloud:443

    # consumer group id
    group.id=kafka-perf

    # SCRAM Properties
    security.protocol=SASL_SSL
    sasl.mechanism=SCRAM-SHA-512
    sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule
required username="<username>" password="<password>";
    # TLS Properties
    ssl.protocol=TLSv1.2
    ssl.truststore.location=/certs/cluster/ca.p12
    ssl.truststore.password= <truststore-password>
    ssl.truststore.type=PKCS12
```

Producer and Consumer Kubernetes Jobs

```
#
# Start of consumer job
#
---
apiVersion: batch/v1
kind: Job
metadata:
  name: perf-consumer
spec:
  template:
    metadata:
      labels:
        app: my-consumer
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: my-consumer
      containers:
      - name: kafka-perf
        image: cp.icr.io/cp/ibm-eventstreams-
kafka@sha256:bd6ce9a4d2ee9aa44cc76efb5a1625122155b082b1c93a4fb8ed55aadd79f2b1
        command:
        - "/opt/kafka/bin/kafka-consumer-perf-test.sh"
        - "--topic"
        - "perf-test"
        - "--messages"
        - "20000000"
        - "--group"
        - "kafka-perf"
        - "--consumer.config"
        - "/config/consumer.properties"
        - "--timeout"
        - "120000"
        - "--bootstrap-server"
        - "prod-3-brokers-kafka-bootstrap-es.itzroks-06000081s7-o0sfdl-
6ccd7f378ae819553d37d5f2ee142bd6-0000.eu-gb.containers.appdomain.cloud:443"
        volumeMounts:
        - mountPath: /certs/cluster
          name: cluster-ca
          readOnly: true
        - mountPath: /certs/mtls-user
          name: mtls-user
          readOnly: true
        - mountPath: /config
          name: consumer-properties
          readOnly: true
      volumes:
        - name: cluster-ca
          secret:
            defaultMode: 420
            items:
            - key: ca.crt
              path: ca.crt
            - key: ca.p12
              path: ca.p12
            - key: ca.password
              path: ca.password
            secretName: prod-3-brokers-cluster-ca-cert
        - name: mtls-user
          secret:
            defaultMode: 420
            items:
            - key: ca.crt
              path: ca.crt
            - key: user.p12
              path: user.p12
            secretName: mtls-user
        - name: consumer-properties
          configMap:
            defaultMode: 420
            name: consumer-properties
      restartPolicy: Never
      resources:
        limits:
          cpu: "2"
          memory: 8096Mi
  backoffLimit: 1
  completions: 18
  parallelism: 18
---
#
#   Start of producer job
#
#
#   Start of producer job
#
apiVersion: batch/v1
kind: Job
metadata:
  name: perf-producer
spec:
  template:
    metadata:
      labels:
        app: my-producer
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: my-producer
      containers:
      - name: kafka-perf
        image: cp.icr.io/cp/ibm-eventstreams-
kafka@sha256:bd6ce9a4d2ee9aa44cc76efb5a1625122155b082b1c93a4fb8ed55aadd79f2b1
        command:
        - "/opt/kafka/bin/kafka-producer-perf-test.sh"
        - "--topic"
        - "perf-test"
        - "--num-records"
        - "20000000"
        - "--record-size"
        - "128"
        - "--throughput"
        - "-1"
        - "--producer.config"
        - "/config/producer.properties"
        volumeMounts:
        - mountPath: /certs/cluster
          name: cluster-ca
          readOnly: true
        - mountPath: /certs/mtls-user
          name: mtls-user
          readOnly: true
        - mountPath: /config
          name: producer-properties
          readOnly: true
      volumes:
        - name: cluster-ca
          secret:
            defaultMode: 420
            items:
            - key: ca.crt
              path: ca.crt
            - key: ca.p12
              path: ca.p12
            - key: ca.password
              path: ca.password
            secretName: prod-3-brokers-cluster-ca-cert
        - name: mtls-user
          secret:
            defaultMode: 420
            items:
```

```
          - key: ca.crt
            path: ca.crt
          - key: user.p12
            path: user.p12
          secretName: mtls-user
      - name: producer-properties
        configMap:
          defaultMode: 420
          name: producer-properties
    restartPolicy: Never
    resources:
      limits:
        cpu: "2"
        memory: 8096Mi
backoffLimit: 1
completions: 6
parallelism: 6
```