



**PLATFORM DEEP DIVE / Plugins / Plugins setup guides / customer-management-plugin-configuration**

# Contents

- PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Customer management plugin setup
  - Infrastructure Prerequisites:
    - Elastic Search
    - Postgres database
  - Configuration
    - Authorization configuration
    - Datasource configuration
    - Elastic search configuration
    - Kafka configuration
    - Logging

## PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Customer management plugin setup

### Infrastructure Prerequisites:

The Customer management plugin is available as a docker image so we need to configure:

### Elastic Search

In order to install elasticsearch instance Elastic Cloud on Kubernetes (ECK) can be used.

Use ECK quickstart to deploy CRDs and create elasticsearch instances:

Elasticsearch instance:

```
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: elasticsearch-flowx
  namespace: elastic-system
spec:
  version: 7.9.3
  updateStrategy:
    changeBudget:
      maxSurge: 3
      maxUnavailable: 1
  nodeSets:
    # 3 dedicated master nodes
    - name: master
      count: 3
      config:
        node.master: true
        node.data: false
        node.ingest: false
        node.remote_cluster_client: false
        # this allows ES to run on nodes even if their
        vm.max_map_count has not been increased, at a performance
        cost

        # node.store.allow_mmap: false
  podTemplate:
    spec:
      initContainers:
        - name: sysctl
          securityContext:
            privileged: true
```

```
        command: ['sh', '-c', 'sysctl -w
vm.max_map_count=262144']
    - name: install-plugins
      command:
        - sh
        - -c
        - |
          bin/elasticsearch-plugin install --batch
repository-gcs
containers:
  - name: elasticsearch
    resources:
      limits:
        memory: 6Gi
        cpu: 2
      requests:
        memory: 2Gi
        cpu: 1
    env:
      - name: ES_JAVA_OPTS
        value: "-Xms2g -Xmx2g"
      - name: READINESS_PROBE_TIMEOUT
        value: "10"
    readinessProbe:
      exec:
        command:
          - bash
          - -c
          - /mnt/elastic-internal/scripts/readiness-
probe-script.sh
      failureThreshold: 3
      initialDelaySeconds: 10
      periodSeconds: 12
      successThreshold: 1
      timeoutSeconds: 12
```

```
    affinity:
      podAntiAffinity:

preferredDuringSchedulingIgnoredDuringExecution:
  - weight: 100
    podAffinityTerm:
      labelSelector:
        matchLabels:
          elasticsearch.k8s.elastic.co/cluster-
name: elasticsearch-flowx
      topologyKey: kubernetes.io/hostname
      # request 2Gi of persistent data storage for pods in
      # this topology element
    volumeClaimTemplates:
      - metadata:
          name: elasticsearch-data
        spec:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 5Gi
          storageClassName: standard
      # 3 ingest-data nodes
      - name: ingest-data
        count: 3
        config:
          node.master: false
          node.data: true
          node.ingest: true
          # this allows ES to run on nodes even if their
          # vm.max_map_count has not been increased, at a performance
          # cost
          # node.store.allow_mmap: false
    podTemplate:
```

```
spec:
  initContainers:
    - name: sysctl
      securityContext:
        privileged: true
        command: ['sh', '-c', 'sysctl -w
vm.max_map_count=262144']
      containers:
        - name: elasticsearch
          resources:
            limits:
              memory: 8Gi
              cpu: 2
            requests:
              memory: 4Gi
              cpu: 1
          env:
            - name: ES_JAVA_OPTS
              value: "-Xms2g -Xmx2g"
          affinity:
            podAntiAffinity:

preferredDuringSchedulingIgnoredDuringExecution:
  - weight: 100
    podAffinityTerm:
      labelSelector:
        matchLabels:
          elasticsearch.k8s.elastic.co/cluster-
name: elasticsearch-flowx
      topologyKey: kubernetes.io/hostname
    # nodeSelector:
    #   diskType: ssd
    #   environment: production
    # request 2Gi of persistent data storage for pods in
    this topology element
```

```
volumeClaimTemplates:
- metadata:
  name: elasticsearch-data
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 20Gi
    storageClassName: standard
```

(Optional) Kibana instance:

```
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
  name: kibana-flowx
  namespace: elastic-system
spec:
  version: 7.9.3
  count: 1
  elasticsearchRef:
    name: elasticsearch-flowx
    namespace: elastic-system
  config:
    elasticsearch.requestHeadersWhitelist:
    - authorization
  podTemplate:
    spec:
      containers:
      - name: kibana
        resources:
          requests:
```

```
memory: 1Gi
cpu: 0.5
limits:
  memory: 3Gi
  cpu: 2
```

The index used by customer management plugin should be created.

## Postgres database

This plugin can work without this database, it will not store the audit data.

### Basic Postgres configuration

```
crmdb:
  existingSecret: {{secretName}}
  metrics:
    enabled: true
    service:
      annotations:
        prometheus.io/port: {{prometheus port}}
        prometheus.io/scrape: "true"
      type: ClusterIP
  serviceMonitor:
    additionalLabels:
      release: prometheus-operator
    enabled: true
    interval: 30s
    scrapeTimeout: 10s
  persistence:
    enabled: true
    size: 4Gi
  postgresqlDatabase: {{postgres databaseName}}
```



```
postgresqlUsername: {{postgres user}}
resources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 200m
    memory: 256Mi
service:
  annotations:
    fabric8.io/expose: "false"
```

## Configuration

### Authorization configuration

The following variables need to be set in order to connect to the identity management platform:

SECURITY\_OAUTH2\_BASE\_SERVER\_URL

SECURITY\_OAUTH2\_CLIENT\_CLIENT\_ID

SECURITY\_OAUTH2\_REALM

### Datasource configuration

To store audit for searches this plugins use a postgres database.

The following configuration details need to be added using environment variables:

`SPRING_DATASOURCE_URL``SPRING_DATASOURCE_USERNAME``SPRING_DATASOURCE_PASSWORD`

You will need to make sure that the user, password, connection link and db name are configured correctly, otherwise you will receive errors at start time.

If you are going to use a database to store the audit, you can use the built-in script to maintain the database schema.

## Elastic search configuration

The connection to elastic search cluster is done over https using the elastic search api. To connect to the it you will need to configure the connection details and index use to store customers.

```
elasticsearch:
  ssl: false
  nodes:
    -
      hostname: ${ELASTICSEARCH_HOST}
      port: ${ELASTICSEARCH_PORT}
      scheme: ${ELASTICSEARCH_HTTP_SCHEME}
  user: ${ELASTICSEARCH_USER}
  password: ${ELASTICSEARCH_PASSWORD}
  customer-index: ${ELASTICSEARCH_CUSTOMER_INDEX}
  size: 10
```

## Kafka configuration

The following Kafka related configurations can be set by using environment variables:

`SPRING_KAFKA_BOOTSTRAP_SERVERS` - address of the Kafka server

`SPRING_KAFKA_CONSUMER_GROUP_ID` - group of consumers

`KAFKA_CONSUMER_THREADS` - the number of Kafka consumer threads

`KAFKA_AUTH_EXCEPTION_RETRY_INTERVAL` - the interval between retries after `AuthorizationException` is thrown by `KafkaConsumer`

`KAFKA_MESSAGE_MAX_BYTES` - this is the largest size of the message that can be received by the broker from a producer.

Each action available in the service corresponds to a Kafka event. A separate Kafka topic must be configured for each use-case.

### CAUTION

The Engine is listening for messages on topics with names of a certain pattern, make sure to use correct outgoing topic names when configuring the documents plugin.

Needed topics:

`KAFKA_TOPIC_CUSTOMER_SEARCH_IN`

`KAFKA_TOPIC_CUSTOMER_SEARCH_OUT`

### CAUTION

In order to match a request made to the customer management plugin, the engine will have to send the process id on a Kafka header.

## Logging

The following environment variables could be set in order to control log levels:

`LOGGING_LEVEL_ROOT` - root spring boot microservice logs

`LOGGING_LEVEL_APP` - app level logs

**Was this page helpful?**