Genesys Predictive Routing Deployment and Operations Guide

Deploy Data Loader on EKS

2/5/2026

Contents

- Administrator

Feature coming soon! Deploy Data Loader in Docker containers hosted on Amazon Elastic Kubernetes Service (EKS).

Related documentation:

-

This page describes the process of installing/configuring/starting the Data Loader application in a Kubernetes environment, specifically AWS EKS. The same instructions are applicable to Azure or GCP deployments with appropriate modifications as needed.

Currently only AWS EKS has been tested and certified. Refer to AWS documentation for creating and managing clusters in your environment.

The compatible Data Loader image version for use in EKS is 9.0.020.00 (or higher). This can be requested from Genesys Software Delivery team. You can install the DL image in a private repository such as JFrog for use in the EKS cluster .

Infrastructure Assumptions

- AWS EKS Cluster
- Nodes
  - To run DL pod, the worker nodes should have enough memory and CPU count to match the recommended resource limit for DL pod(Memory: 16 GB & CPU count: 4)
- Networking
  - Config Server, GIM, and GPR should be accessible from the worker node where DL pod is running.
  - DL deployment in EKS will be exposed as service through ELB(Load Balancer).
  - LoadBalancer URL will be used as host in Config Server for the DL application.
- Persistent Storage
  - To run custom SQL files which is used to fetch the data from GIM.
  - To preserve the log files of the DL application.

Set up the Environment

Create a namespace for Data Loader using the following command:

```
kubectl apply -f namespace.json
```

The **namespace.json** defines how the namespace is to be created. Modify the following sample to suit your environment:

```
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "dataloader",
    "labels": {
      "name": "dataloader"
    }
  }
}
```

> **Important**
>
> - Replicas are not required for Data Loader.
> - Ensure that the resources have a 4 CPU count and at least 16 GB memory available.
> - In AWS EKS, namespaces can be used in a multi-tenant environment.

Before using the data loader image, create the secrets for the image using the following command:

```
kubectl create secret docker-registry jfrogcred --docker-server= pureengage-docker-
staging.jfrog.io --docker-username= --docker-password= --docker-email= -n dl
```

Deploy Data Loader

Ensure that the Data Loader image is available in the cluster or pulled from the repo defined in the **deployment.yaml** file.

Create a Config Map file

The **ConfigMap.yaml** file stores the environment variables used by the Data Loader container. These values are read from the **deployment.yaml** file when the deployment is created.

To create the **ConfigMap.yaml** file, run

```
kubectl apply -f DL_ConfigMap.yaml -n dataloader
```

OR

```
kubectl apply -f DL_ConfigMap.yaml --namespace=dataloader
```

A sample **ConfigMap.yaml** file:

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: dl-primary-config
  namespace: dl # Assumption is namespace dl is already created
data:
  JVMPARAMS: "-server -Xmx2g -Xms2g -Xss512k"
  START_DL: "false"
  CONFIG_HOST: "10.52.86.42"
  CONFIG_PORT: "8888"
  CONFIG_DL_APP_NAME: "DataLoader-Alpha"
  LCA_PORT: "4999"
```

Configure Persistent Storage

A persistent storage is required for storing the log files and read SQL scripts for processing CSV files. Persistent Volumes are resources in the cluster and Persistent Volume Claims are requests for those resources and can also act as claim checks for the resource.

The following configuration will map the local volume to the Kubernetes cluster.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: dl-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/dl_data"
```

The following sample will be used during deployment to mount the volume in the cluster.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: dl-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

> **Important**
>
> The Persistent Volume Claim will use the persistent volume for writing the logs and reading SQL scripts.

Deploying the Data Loader application

The data Loader application can be deployed using the standard Kubernetes deployment mode and

can be used to control the Data Loader container pod in t he AWS EKS cluster to manage and scale the Data Loader application.

To deploy the Data Loader application using a YAML file, run

```
kubectl apply -f DL.yaml -n dataloader
```

A sample DL.yaml file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dataloader-primary
  namespace: dl
  labels:
    app: dataloader-primary
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dataloader-primary
  template:
    metadata:
      labels:
        app: dataloader-primary
    spec:
      volumes: # Assumption is persistent storage claim dl-pv-claim is already created for
storing the sql files and logs
      - name: dl-pv-storage
        persistentVolumeClaim:
          claimName: dl-pv-claim
      imagePullSecrets:
      - name: jfrogcred # Assumption is secret jfrogcred is already created to pull the image
      containers:
      - name: dataloader-primary
        image: pureengage-docker-staging.jfrog.io/gpr/ai-data-loader:9.0.020.00
        resources:
          limits:
            cpu: "4"
            memory: "8Gi"
        volumeMounts:
        - mountPath: /dl/mount # Based on the mount path, dl application should be configured
to write the log files or read the sql files in the mount path.
          name: dl-pv-storage
        ports:
        - containerPort: 8091
        - containerPort: 4999
        env:
        - name: JVMPARAMS
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: JVMPARAMS
        - name: START_DL
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: START_DL
        - name: CONFIG_HOST
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
```

```
            key: CONFIG_HOST
        - name: CONFIG_PORT
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: CONFIG_PORT
        - name: CONFIG_DL_APP_NAME
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: CONFIG_DL_APP_NAME
        - name: LCA_PORT
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: LCA_PORT
```

Configure High Availabliity

Setting up a load balancer

To set up a load balancer, modify the following configuration in the YAML file,

```
apiVersion: v1
kind: Service
metadata:
  name: dl-primary-service
  namespace: dl
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
    service.beta.kubernetes.io/aws-load-balancer-internal: "true"
spec:
  loadBalancerSourceRanges:
  - 10.52.0.0/17
  - 172.20.0.0/17
  type: LoadBalancer
  selector:
    app: dataloader-primary
  ports:
    - protocol: "TCP"
      name: "lca"
      port: 4999
      targetPort: 4999
    - protocol: "TCP"
      name: "dl"
      port: 8091
      targetPort: 8091
```

> **Important**
>
> Autoscaling is not supported.