Genesys Info Mart Private Edition Guide

Deploy GIM

7/13/2025

Contents

Learn how to deploy GIM (GIM) into a private edition environment.

Related documentation:

-
-
-
-

RSS:

- For private edition

Assumptions

- The instructions on this page assume you are deploying the service in a service-specific namespace, named in accordance with the requirements on Creating namespaces. If you are using a single namespace for all private edition services, replace the namespace element in the commands on this page with the name of your single namespace or project.
- Similarly, the configuration and environment setup instructions assume you need to create namespace-specific (in other words, service-specific) secrets. If you are using a single namespace for all private edition services, you might not need to create separate secrets for each service, depending on your credentials management requirements. However, if you do create service-specific secrets in a single namespace, be sure to avoid naming conflicts.

> **Important**
>
> Make sure to review Before you begin GIM deployment for the full list of prerequisites required to deploy GIM, including creation of the Info Mart database and, if applicable, S3-compatible storage for your exported data (see (see Create object storage).

Set up your environment

To prepare your environment for the deployment, complete the steps in this section for:

- GKE
- AKS

GKE environment setup

1. Ensure that the gcloud CLI and required Helm version are installed on the host where you will run the deployment.
2. Using the CLI, log in to the GKE cluster from the host where you will run the deployment:

   ```
   gcloud container clusters get-credentials
   ```

3. If the cluster administrator has not already done so, create a new namespace `gim` for GIM:
   - Create a JSON file specifying the namespace metadata. For example, **create-gim-namespace.json**:

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

   - Execute the following command to create the `gim` namespace:

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

   - Confirm namespace creation:

     ```
     kubectl describe namespace gim
     ```

4. Create the pull secret for the image registry.
   - This step defines a secret so that Kubernetes can authenticate your image repository and pull artifacts from it. The repository is represented as `docker-registry` in the system. For information about downloading artifacts from the repository, see Downloading your Genesys Multicloud CX containers.
   - When you configure the GIM service, you will reference the registry pull secret as a Helm chart override; see GIM Helm chart overrides.

     ```
     kubectl create secret docker-registry  --docker-server= --docker-username= --docker-
      password= --docker-email= -n gim
     ```

AKS environment setup

1. Ensure that the Azure CLI and required Helm version are installed on the host where you will run the deployment.
2. Using the CLI, log in to the Azure cluster from the host where you will run the deployment.

   ```
   az aks get-credentials --resource-group  --name  --admin
   ```

3. If the cluster administrator has not already done so, create a new namespace `gim` for GIM:
   - Create a JSON file specifying the namespace metadata; for example, **create-gim-namespace.json** as in the following example:

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

- Execute the following command to create the `gim` namespace:

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

- Confirm namespace creation:

```
kubectl describe namespace gim
```

4. Create the pull secret for the image registry.
   - This step defines a secret so that Kubernetes can authenticate your image repository and pull artifacts from it. The repository is represented as `docker-registry` in the system. For information about downloading artifacts from the repository, see Downloading your Genesys Multicloud CX containers.
   - When you configure the GIM service, you will reference the registry pull secret as a Helm chart override; see GIM Helm chart overrides.

```
kubectl create secret docker-registry  --docker-server= --docker-username= --docker-
 password= --docker-email= -n gim
```

Deploy

Execute the following command to install GIM:

```
helm upgrade --install gim  -f  -n gim
```

Execute the following command to install GIM monitoring:

```
helm upgrade --install gim-monitoring  -n gim
```

When the GIM service starts, it connects to the Info Mart database and checks if the GIM schema has been initialized. If the schema has not been initialized, the service runs a script to initialize the GIM schema.

Validate the deployment

You can consider GIM deployment successful when the pod is running and in Ready state. Genesys Info Mart does not report the Ready state for pods until internal health checks are satisfied and the pods are operational. For example, GIM does not report Ready until the first transformation job has completed successfully. In other words, if the GIM pod is running and in Ready state, it means that GIM successfully started, connected to Kafka and the Info Mart database, initialized the Info Mart

database, and completed the first ETL cycle.

You can use standard `kubectl` commands like `list` and `get` to verify the successful deployment and readiness status of the Kubernetes objects, including connection to the database.

However, from a functional point of view, you cannot validate deployment of the GIM service and database unless GCA and GSP have been deployed as well. Do not expect consistent data until all three Genesys Info Mart services are up and running. For more details about functional checks you can perform to validate GIM deployment, see the equivalent validation section on the Deploy GIM Stream Processor page.