Genesys Info Mart Private Edition Guide

Deploy GIM Stream Processor

4/16/2026

Contents

- 1 Assumptions
- 2 Set up your environment
  - 2.1 GKE environment setup
  - 2.2 AKS environment setup
- 3 Deploy the GSP
- 4 Validate the deployment

Learn how to deploy GIM Stream Processor (GSP) 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 GSP deployment for the full list of prerequisites required to deploy GSP, including provisioning S3-compatible storage (see Before you begin GSP deployment).

Set up your environment

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

- 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 on 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 `gsp` for GSP.
   - Create a JSON file that specifies the namespace metadata. As an example, the file **create-gsp-namespace.json.** has the following JSON

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

   - Create the namespace. The following example applies the **create-gsp-namespace.json** file to create the `gsp` namespace.

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

   - Confirm that the namespace was successfully created.

     ```
     kubectl describe namespace gsp
     ```

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 GSP, you will reference the registry pull secret as a Helm chart override; see GSP Helm chart overrides.

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

5. Create the Kafka secret.
   - Kafka requires a separate access credential.
   - This credential is represented as `kafka-secrets` in the system, as in the following syntax.

     ```
     kubectl create secret generic kafka-secrets --from-literal=kafka-
     secrets={\"bootstrap\":\"\", \"username\":\"gsp\",\"password\":\"\"} -n gsp
     ```

     For example:

     ```
     kubectl create secret generic kafka-secrets --from-literal=kafka-
     secrets={\"bootstrap\":\"infra-kafka-cp-
     kafka.infra.svc.cluster.local:9092\",\"username\":\"gsp\",\"password\":\"kafka-
     password\"} -n gsp
     ```

Note: Kafka can be deployed without authentication, as in the following example.

```
kubectl create secret generic kafka-secrets --from-literal=kafka-
 secrets={\"bootstrap\":\"\"} -n gsp
```

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 cluster on 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 `gsp` for GSP.
   - Create a JSON file that specifies the namespace metadata. As an example, the file **create-gsp-namespace.json** has the JSON shown below.

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

   - Create the namespace. The following example applies the **create-gsp-namespace.json** file to create the `gsp` namespace.

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

   - Confirm that the namespace was successfully created.

     ```
     kubectl describe namespace gsp
     ```

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 GSP, you will reference the registry pull secret as a Helm chart override; see GSP Helm chart overrides.

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

5. Create the Kafka secret.
   - Kafka requires a separate access credential.
   - This credential is represented as `kafka-secrets` in the system, as in the following syntax.

     ```
     kubectl create secret generic kafka-secrets --from-literal=kafka-
      secrets={\"bootstrap\":\"\", \"username\":\"gsp\",\"password\":\"\"} -n gsp
     ```

For example:

```
kubectl create secret generic kafka-secrets --from-literal=kafka-
 secrets={\"bootstrap\":\"infra-kafka-cp-
 kafka.infra.svc.cluster.local:9092\",\"username\":\"gsp\",\"password\":\"kafka-
 password\"} -n gsp
```

Note: Kafka can be deployed without authentication, as in the following example.

```
kubectl create secret generic kafka-secrets --from-literal=kafka-
 secrets={\"bootstrap\":\"\"} -n gsp
```

Deploy the GSP

After the environment has been set up, deploying the GSP is a matter of executing the following command:

```
helm install gsp  -f  -n gsp
```

Validate the deployment

You can consider GSP 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. You can use standard `kubectl` commands like `list` and `get` to verify the successful deployment and readiness status of the Kubernetes objects.

However, from a functional point of view, you cannot validate GSP deployment unless GCA and GIM have been deployed as well. Do not expect consistent data until all three Genesys Info Mart services are up and running. When all three services have been deployed:

1. Make a few test calls employing different routing strategies under different scenarios, and verify that all the calls are correctly captured in the Info Mart database. For example:
   - The calls appear in the interaction-related tables.
   - The calls have been correctly assigned to agents and queues.
2. Review the logs to verify no errors.
3. Monitor the operations dashboard to verify that the services report their status as Ready, and pods are not continually restarting.