Genesys Customer Experience Insights Deployment Guide

Installing Genesys CX Insights - Kubernetes using descriptors

5/8/2025

Contents

Installing Genesys CX Insights - Kubernetes using descriptors

> **Important**
>
> In keeping with Genesys' commitment to diversity, equality, and inclusivity, beginning with release 9.0.019.01, some pod names are changed; this document refers to "gcxi-primary" and "gcxi-secondary" pods. In release 9.0.019.00 and earlier, these pods were named "gcxi-master" and "gcxi-slave".

This page describes the steps required to deploy Genesys Customer Experience Insights (Genesys CX Insights) with Kubernetes, using descriptors, which is the supported and recommended scenario for deploying Genesys CX Insights in most production environments in release 9.0.015 and earlier.

Beginning with release 9.0.016.02, Genesys supports deployment of Kubernetes using Helm, which is recommended for new Kubernetes deployments. This deployment method is suitable for production environments; for other deployment options, see Choose a deployment type and Prerequisites.

Before You Begin

Before you complete the steps on this page, prepare the environment as described in Before you install Genesys CX Insights. In addition:

1. Acquire the Genesys CX Insights installation package

Ensure that you have the latest Genesys CX Insights 9.0 installation packages (IP); talk to your Genesys representative for information about where to download the installation packages.

Installation packages for GCXI

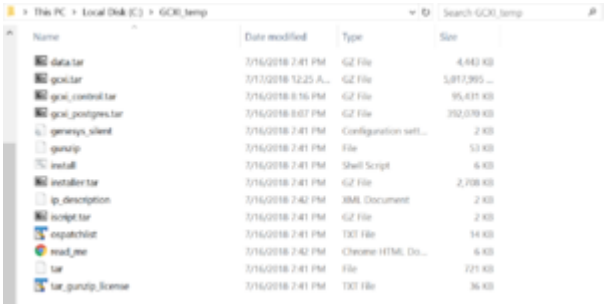
Component	IP / File	tar files*
CustExInsights — Genesys Customer Experience Insights	Docker container (Docker Linux platform) **IP_CustExpInsights_9000XXX_ENU_dockerlinux.zip** (where XXX is the latest release number.)	**gcxi.tar.gz** — contains the **gcxi** Docker image, which contains a fully installed Microstrategy Server 10.x (the latest supported release of MicroStrategy: 11.2.1, for example.). This container provides a *stateless* deployment, where project data (reports, users, and other objects) is stored separately in a MicroStrategy *meta* database. This image is used for production deployments.
	Regular Linux IP (Linux platform)	**data.tar.gz** — contains the various YAML files

	IP_CustExpInsights_9000XXX_ENU_linux.tar.gz (where XXX is the latest release number.)	(Kubernetes script files), such as gcxi.yaml, gcxi-postgres-yaml, and gcxi-init.yaml, and the gcxi.properties file (files which you must edit as part of the deployment procedure), PostgreSQL database dump with MicroStrategy meta-data database for GCXI project, and other files needed for GCXI
CustExInsightOps — Genesys Customer Experience Insights Ops	Docker container (Docker Linux platform) **IP_CustExpInsightsOPS_9000XXX_ENU_dockerlinux.zip** (where XXX is the latest release number.)	**gcxi_control.tar.gz** — contains the **gcxi_control** Docker image, which is used for deployment and configuration of the GCXI solution.
CustExInsightDB — Genesys Customer Experience Insights DB (Discontinued beginning with GCXI release 9.0.019.01)	Docker container (Docker Linux platform) **IP_CustExpInsightsDB_9000XXX_ENU_dockerlinux.zip** (where XXX is the latest release number.)	**gcxi_postgres.tar.gz** — contains the **gcxi_postgres** image, which contains a PostgreSQL database server with GCXI MicroStrategy Project, Meta data, and History databases pre-deployed. This image is discontinued beginning with GCXI release 9.0.019.01
MSSecEntPltf64 — MicroStrategy Secure Enterprise Platform for Windows	MicroStrategy software for Windows (server and client / editing tools) **MicroStrategy_XXX_IntelligentEnterprise_Windows_XXX.zip** (where XX is the current MicroStrategy release. For example, **MicroStrategy_11.3_IntelligentEnterprise_Windows_11.3.0560.0066.zip**.)	MicroStrategy_Secure_Enterprise_Platform_11_3_Win/cpe1705/PI
MSWrkstn — MicroStrategy Workstation	MicroStrategy Workstation software for Windows **workstation-win-ent_XXX.zip** (where XXX is the current MicroStrategy release. For example, **workstation-win-ent_11.3.63.zip**.)	MicroStrategy_Workstation_11.3.63/ The MicroStrategy Workstation package is available beginning with GCXI release 100.0.029.0000
*Important: In some releases, the names of the container images in the installation package differ from the description in the **Installation packages for GCXI** table. In these scenarios, rename the container images as described in the table **Renaming the images**:		

Renaming the images (if your release requires it)

Copy this file from this folder to a convenient location on your local hard drive (for example `C:\GCXI_temp`):	Rename it as:
CustExpInsights ... **dockerlinux**... **9.0.010.04.tar.gz**	**gcxi.tar.gz**
CustExpInsightsDB ... **9.0.010.04.tar.gz** (This image is discontinued beginning with GCXI release 9.0.019.01)	**gcxi_postgres.tar.gz**
CustExpInsightsOps ... **9.0.010.04.tar.gz**	**gcxi_control.tar.gz**

Note that Reporting and Analytics Aggregates (RAA) files are also available in the same location (**Reporting_and_Analytics_Aggregates_G231_850XXXX_ENU_ISO**). See the Reporting and Analytics Aggregates documentation for more information about deploying RAA.



The installation package contents

2. Gather information about data sources

For Genesys CX Insights to produce meaningful reports, you must have installed and properly configured both Genesys Info Mart and Reporting and Analytics Aggregates (RAA):

- Genesys Info Mart release 8.5 database — The Genesys Info Mart documentation describes how to deploy and configure Genesys Info Mart, including information about hardware sizing requirements to support Gensys Info Mart. Genesys CX Insights can provide meaningful reports only if the Info Mart database is regularly populated by a Genesys Info Mart 8.5 application. Genesys Info Mart must be properly configured and installed before Genesys CX Insights runs the aggregation process (RAA). Refer to the *Genesys Info Mart Deployment Guide* or the *Genesys Migration Guide* for information that pertains to configuring, installing, or upgrading Genesys Info Mart.

- Reporting and Analytics Aggregates (RAA) — The RAA documentation describes how to deploy RAA, and how to configure the aggregation process. You must have available all relevant Genesys Info Mart information, including the RDBMS type (Microsoft SQL Server, PostgreSQL, Oracle), hostname, and user credentials.

3. Decide how to handle the meta database

Choose whether to deploy an external PostgreSQL server for the *meta* database:

- Deploying with an external meta database — Requires that you create an external PostgreSQL server on which the MicroStrategy meta database resides.
- Deploying the pre-packaged meta database — Uses a pre-packaged meta database, so that you do not need to deploy or manage a separate PostgreSQL server. This option uses the standard PostgreSQL container. If you choose this option, the PostgreSQL container must have access to its data volume.

Deploying the containers

Use the steps in this section to deploy the Docker containers. Note that Genesys does not ship Docker or Kubernetes as a part of Genesys CX Insights. You must install Docker in your environment before you can load the Genesys CX Insights containers; see Before you install Genesys CX Insights. Install Docker according to the instructions on the Docker site. A Docker deployment provides a complete self-contained environment, so that you do not need to manually configure ports or address compatibility issues.

Procedure: Load Docker images from tar.gz archives

Purpose: Use the steps in this procedure to prepare the Docker containers.

Steps

Complete the following steps on EACH machine, except where noted otherwise:

1. Copy the docker containers (gcxi_control.tar.gz and gcxi.tar.gz) onto each machine in the cluster.
2. Log in as root, or execute the following command to switch to root:

   ```
   sudo -i bash
   ```
3. On each machine in the cluster, execute the following command:

   ```
   docker load < gcxi_control.tar.gz
   ```
4. On each machine in the cluster, execute the following command:

   ```
   docker load < gcxi.tar.gz
   ```

Procedure: Retag Images

Purpose: Images can contain tagging strings that cause installation errors. Use the steps in this procedure to ensure that images have proper tagging.

Prerequisites

- In release 9.0.011 and later, this procedure applies to all image files, whether downloaded from a repository, or from tar.gz files.

Steps

1. Execute the following command to verify that the images loaded correctly:

   ```
   docker images
   ```

   The console lists the installed Docker images.

   ```
   $ docker images
   REPOSITORY                                                TAG                 IMAGE ID            CREATED             SIZE
   pureengage-docker-production.jfrog.io/gcxi/gcxi_control   9.0.013.01          761b48dfa69a        6 weeks ago         1.32GB
   pureengage-docker-production.jfrog.io/gcxi/gcxi           9.0.013.01          e7a7216f2f2f        6 weeks ago         11.7GB
   pureengage-docker-production.jfrog.io/gcxi/gcxi_postgres  9.0.013.01          068b8c6ba06c        6 weeks ago         3.53GB
   ```

2. Execute the following commands to retag each of the images:

   ```
   docker tag <REPOSITORY>/gcxi:<RELEASE> gcxi:<RELEASE>
   ```

   ```
   docker tag <REPOSITORY>/gcxi_postgres:<RELEASE> gcxi_postgres:<RELEASE>
   ```

   ```
   docker tag <REPOSITORY>/gcxi_control:<RELEASE> gcxi_control:<RELEASE>
   ```

   where:

   <REPOSITORY> is the identifier for the repository from which you downloaded the files.

   <RELEASE> is the a string corresponding to the release you are installing (such as 9.0.014.02),

   For example:

   ```
   docker tag pureengage-docker-production.jfrog.io/gcxi/gcxi:9.0.014.02 gcxi:9.0.014.02
   ```

   ```
   docker tag pureengage-docker-production.jfrog.io/gcxi/gcxi_postgres:9.0.014.02 gcxi_postgres:9.0.014.02
   ```

   ```
   docker tag pureengage-docker-production.jfrog.io/gcxi/gcxi_control:9.0.014.02 gcxi_control:9.0.014.02
   ```

3. Execute the following command to verify that the images loaded correctly, and have correct tagging:

   ```
   docker images
   ```

   The console lists the installed Docker images. Compare the result to the figure **Docker Images**; each image must have a name in the REPOSITORY column *with no preceding path*, and a value in the TAG column that corresponds to the release. Note that each image appears twice in the list, this is expected behavior, because each one has two tags.

Procedure: Install the CustExInsights component

Purpose: Use the steps in this procedure to install the Genesys Customer Experience Insights Linux (CustExInsights) component. This procedure applies to release 9.0.009 and later.

Steps

Complete the following steps on the PRIMARY machine:

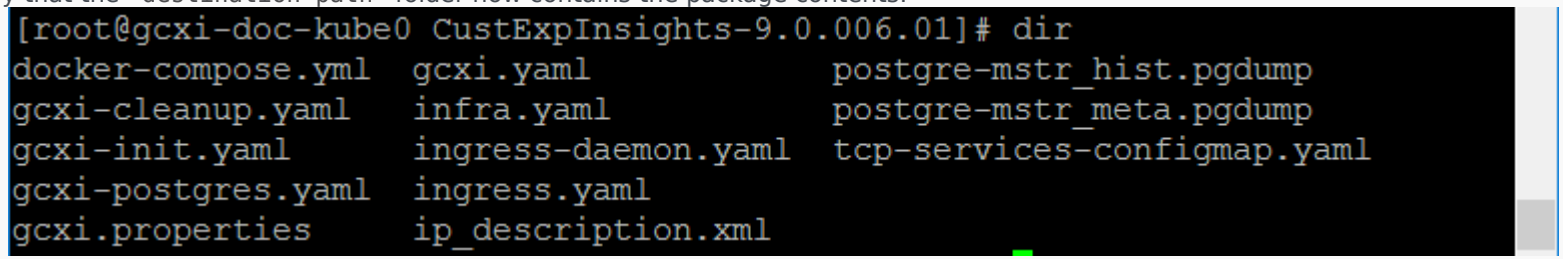
1. Download the CustExpInsights Installation Package (IP), extract its contents, and copy the `\9.0.00x.0x\linux\ip` subfolder to a directory on the PRIMARY machine.
2. Use the following commands to install the software automatically:
   1. Execute the following command to set permissions on the installation script:

      ```
      chmod a+x install.sh
      ```

   2. Execute the installation script:

```
./install.sh
```

3. Verify that the <destination path> folder now contains the package contents:



Tip

To identify the folder where the Genesys CX Insights files are installed, execute the following command:

```
sudo find -name gcxi.properties
```

which returns the path to the `gcxi.properties` file, such as:

```
/genesys/gcxi/gcxi.properties
```

Procedure: Enter database information in the properties file

Purpose: Use the steps in this procedure to populate the **gcxi.properties** file with information about your Info Mart. This procedure is intended for Genesys CX Insights release 9.0.010 and later.

Steps

On the PRIMARY machine, open the **gcxi.properties** file for editing, and edit it as follows:

Genesys CX Insights database properties

This section of the file provides information about defining data sources. Two methods are supported: DSNDEF variables and an externally mounted **obdc.ini** file (supported in GCXI release 9.0.015 and later) — choose one of the two methods. Alternatively, if you don't populate define a data source (using either DSN variables or an OBDC file, as discussed) Genesys CX Insights uses the demo database provided in the container.

Defining data sources using DSNDEF variables

Beginning with release 9.0.010, Genesys CX Insights supports more than one project, and allows you to define multiple DSNs, as follows:

1. Define a **DSNDEF** variable for each DSN. The name of the variable must consist of the string DSNDEF, following by one or more unique characters, for example DSNDEF1, followed by the relevant properties, in the following format:

   ```
   DSNDEF1=DRV_TYPE=<DRV_TYPE>;GCXI_QENGINE=<GCXI_QENGINE>;DSN_NAME=<DSN Name>;
    DB_TYPE=<DB_TYPE>;DB_TYPE_EX=<DB_TYPE_EX>;HOST=<host>;
    PORT=<PORT>;ORCL_SNAME=<ORCL_SNAME>;ORCL_SID=<ORCL_SID>;
    DB_NAME=<DB_NAME>;LOGIN=<USERNAME>;PASSWORD=<PASSWORD>;ENCODING=<UTF_OPTION>
   ```

   where:

   <**DRV_TYPE**> (OPTIONAL) is a value that controls the MSTR setting for the corresponding Database Connections object. Consider setting this

option if you are troubleshooting with connection to database. The following values are supported: `ODBC`, `JDBC`. Default value: `ODBC`. For example `DRV_TYPE=JDBC`.

<**GCXI_QENGINE**> (OPTIONAL) is a value that controls the MSTR setting for the corresponding Database Connections object. This option works only when DRV_TYPE is configured to `JDBC`. Consider setting this option if you are troubleshooting the performance of project reports. The following values are supported: `ON`, `OFF`. Default value: `OFF`. For example `GCXI_QENGINE=ON`.

<**DSN_NAME**> is the DSN type. The following values are supported: GCXI_GIM_DB or IWD_DB

<**DB_TYPE**> is the RDBMS type. The following values are supported: POSTGRESQL, SQLSERVER, ORCLW

<**DB_TYPE_EX**> is your extended RDBMS type. For an up-to-date list of supported values, open the file **$MSTR_INSTALL_HOME/install/DATABASE.PDS**, and check the **DSSOBJECT** element, **NAME** attribute. For example, the following values are supported at the time of writing, but may vary depending on the release you are installing: 'Microsoft SQL Server 2012' 'Microsoft SQL Server 2014' 'Microsoft SQL Server 2016' 'POSTGRESQL' 'Oracle 12cR2' 'Oracle 18c' 'Oracle 19c'

Note, however, that there is a known issue with `Oracle 11g`; Genesys recommends that you use `Oracle 11gR2` instead.

<**PORT**> is the Genesys Info Mart or iWD RDBMS port number.

<**DB_LOGIN**> is a Microstrategy object with $DSN_NAME.

For Oracle deployments, populate either **ORCL_SID** or **ORCL_SNAME** (not both):

- <**ORCL_SNAME**> is the Oracle Service Name, an alias that is used to remotely connect to the database.
- <**ORCL_SID**> is the Oracle SID, a unique name that identifies the instance / database.

<**DB_NAME**> is the actual Genesys Info Mart or iWD database name (populate this only for Microsoft SQL and PostgreSQL deployments).

<**USERNAME**> is the Genesys Info Mart or iWD administrator user name.

<**PASSWORD**> is the Genesys Info Mart or iWD password.

<**UTF_OPTION**> (OPTIONAL) is the value that controls the MSTR setting for the corresponding Database Connections object in release 9.0.013 and earlier; **UTF_OPTION** is not used in release 9.0.014 and later.

OR

Define data sources using ODBC

Beginning with release 9.0.015, Genesys CX Insights supports the use of a custom external **obdc.ini** file to define the database. To use this method instead of DSN variables, complete the following steps:

1. Define the variable **SKIP_DSN**, to disable automatic DSN configuration. You can define this variable globally or for individual DSNDEF.
2. Create an externally mounted **odbc.ini** file similar to the following:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: gcxi-config-odbc
  namespace: genesys
data:
  DSN_ODBCINI_OVERRIDE: |-
    [ODBC Data Sources]
    GCXI_GIM_DB=MicroStrategy ODBC Driver for PostgreSQL Wire Protocol
    IWD_DB=MicroStrategy ODBC Driver for PostgreSQL Wire Protocol

    [GCXI_GIM_DB]
    ApplicationUsingThreads=8
    AlternateServers=8

    [IWD_DB]
    ApplicationUsingThreads=9
    Description=MicroStrategy ODBC Driver for PostgreSQL Wire Protocol
    FailoverPreconnect=9
```

   This example uses **DSN_ODBCINI_OVERRIDE**. Alternatively, use DSN_ODBCIN instead of DSN_ODBCINI_OVERRIDE to override only the sections [GCXI_GIM_DB] and [IWD_DB] in the **odbc.ini** file (in all pods).
3. Define the following variable in the **configmap yaml** file:

- **DSN_ODBCINI** — define the entire definition for a particular DSN.

MicroStrategy database properties This section of the file provides information about MicroStrategy *meta* and *history* database properties. These are databases where Microstrategy stores internal information, and are created automatically during Genesys CX Insights deployment. You can choose whether to deploy an external PostgreSQL server for the meta database:

- Deploying with an external meta database — Requires that you create an external PostgreSQL server on which the MicroStrategy meta database resides.
- Deploying the pre-packaged meta database — Uses a pre-packaged meta database, so that you do not need to deploy or manage a separate PostgreSQL server. This option uses the standard PostgreSQL container. If you choose this option, the PostgreSQL container must have access to its data volume, so note the following two possible configurations:
  - PostgreSQL data volume is shared across worker nodes. In this case no additional steps needed, and the PostgreSQL container can run on any worker node,
    OR
  - If PostgreSQL data volume resides on a specific node, you must tie the PostgreSQL container execution to this node by completing the following steps:
    1. Choose a worker node for your PostgreSQL container, and apply a label to the node using the command `kubectl label nodes`.
       For example:

       ```
       kubectl label nodes gcxi/role=gcxi-db
       ```

    2. Open the **gcxi-postgres.yaml** file for editing.
    3. Uncomment the `## nodeSelector example ##` section, and edit it as described in the comments, so that **node-selector** is set to **worker node**. Ensure that the label in nodeSelector matches the one you applied to the node.
       For example:

       ```
       gcxi/role: gcxi-db
       ```

       For more information, see Assign Pods to Nodes on the Kubernetes website.

- Postgres containers store database data in a Docker volume, which is a physical directory on the host that is mapped to the container. By default, this directory is `/genesys/gcxi/data`. Prior to starting the container, you can change the directory that is used to store the data by editing the **gcxi-postgres.yaml.** file, and changing the `/genesys/gcxi/data` to another valid path.

> **Important**
>
> In scenarios where you use the pre-packaged meta database, Genesys strongly recommends that you regularly back up the contents of the **Docker volume** directory, as it contains your Genesys CX Insights database data.

Choose one of the following:

- If you plan to use the pre-packaged meta database, leave all the `META*` properties empty.
- If you are using an external PostgreSQL server to host the meta database, complete the following steps:

1. Populate the `META_DB_ADMIN`, `META_DB_ADMINDB`, and `META_DB_ADMINPWD` properties with an existing user name, database name, and password. The user name specified must correspond to an account that has the necessary permissions to create databases and database users, and to assign ownership.
   For example: `META_DB_ADMIN=postgres`, `META_DB_ADMINDB=postgres_db`, and `META_DB_ADMINPWD=postgres_pwd`.
2. Populate the `META_DB_HOST` and `META_DB_PORT` properties host and port where the meta and history databases will be created:

   ```
   META_DB_HOST=<Host name> and META_DB_PORT=<port>
   ```

   where <Host name> is the host name where the external PostgreSQL server is located, and <port number> is the port of the external PostgreSQL server (usually 5432).
3. Populate the `META_DB_LOGIN` and `META_DB_PASSWORD` properties with credentials for a new user account to be created for the MicroStrategy meta database. The deployment routine uses the information you provide to create a new user account for the meta database.
   For example: `META_DB_LOGIN=mstr_meta_kube` and `META_DB_PASSWORD=g1n2s3s4`

4. Populate the `META_HIST_LOGIN` and `META_HIST_PASSWORD` properties with appropriate credentials for the MicroStrategy history database. The deployment routine uses the information you provide to create a user account for the history database.
   For example: `META_HIST_LOGIN=mstr_hist_kube` and `META_HIST_PASSWORD=g1n2s3s4`.

Other properties

This section of the file provides information about other relevant properties:

- Populate the `MSTR_PASSWORD` property with a suitable administrative password (minimum 8 characters, with 1 each of upper case, lower case, and numeric). The deployment routine uses the information you provide to set the administrator password for Microstrategy.
  For example: `MSTR_PASSWORD=Pa55word`. You can change this later, using the steps in Changing administrator passwords.
- Ensure that the **gcxi.properties** > `GCXI_VERSION` property was set correctly by the installer. It must correspond to the release you are installing, for example `GCXI_VERSION=9.0.014.02`.
- Beginning with release 9.0.013, a new variable (`TOMCAT_GCXIROOT=<false|true>`) allows you to optionally redirect the base URL to provide a shorter form of the URL used to access reports, in the form of `http://<server>` instead of `http://<server>:8080/MicroStrategy`.
  To enable redirect, set the container variable to `true`:

  ```
  TOMCAT_GCXIROOT=true
  ```

Procedure: Deploy Genesys CX Insights

Purpose: Use the steps in this procedure to deploy Genesys CX Insights into Kubernetes.

Steps

Complete the following steps on the PRIMARY machine:

1. To create Kubernetes namespace 'genesys', execute the following command:

   ```
   kubectl create -f <destination path>/infra.yaml
   ```

   where <destination path> is the folder in which the Genesys Installation Package (IP) is stored, for example:

   ```
   kubectl create -f /genesys/gcxi/infra.yaml
   ```

2. To set 'genesys' namespace as the default namespace, execute the following command:

   ```
   kubectl config set-context $(kubectl config current-context) --namespace=genesys
   ```

3. To load the variables into Kubernetes, execute the following command:

   ```
   kubectl create configmap gcxi-config --from-env-file=<destination path>/gcxi.properties --namespace genesys
   ```

   where <destination path> is the folder in which the Genesys IP is stored, for example:

   ```
   kubectl create configmap gcxi-config --from-env-file=/genesys/gcxi/gcxi.properties --namespace genesys
   ```

4. If you are using secrets to store data (as discussed in Configure secrets), execute the following command to create the secrets:

   ```
   kubectl create -f <destination path>/gcxi-secrets.yaml
   ```

5. If you are hosting the meta database on an external server, skip this step. If you are using the pre-packaged meta database, complete the following steps:

   1. Execute the following command to start the PostgreSQL database container, which is required so that your GCXI meta database to run in the PostgreSQL container as a part of the Kubernetes cluster:

      ```
      kubectl create -f <destination path>/gcxi-postgres.yaml
      ```

where <destination path> is the folder in which the Genesys IP is stored, for example:

```
kubectl create -f /genesys/gcxi/gcxi-postgres.yaml
```

2. Execute the following command to verify the state of the gcxi-postgres pod:

   ```
   kubectl get pods | grep 'gcxi-postgres*'
   ```

   The pod status should be Running, for example:

   ```
   gcxi-postgres-5cd4d45754-mss6p   1/1       Running     0          6d
   ```

   If it has any other state, wait a few minutes and check again (it may take some time).

6. To create the MicroStrategy meta database, complete the following steps:

   1. Execute the following command to create the database:

      ```
      kubectl create -f <destination path>/gcxi-init.yaml
      ```

      For example:

      ```
      kubectl create -f /genesys/gcxi/gcxi-init.yaml
      ```

      where <destination path> is the folder in which the Genesys IP is stored. Note that this step is required even if you use the pre-packaged PostgreSQL container.

      Troubleshooting Tip: When creating the meta database, if you encounter an error similar to *Could not translate host name "gcxi-postgres" to address*, see the Troubleshooting page.

   2. Execute the following command to verify the state of the gcxi-init pod:

      ```
      kubectl get pods | grep 'gcxi-init*'
      ```

      The pod status should be Completed, for example:

      ```
      gcxi-init-l4b4x                 0/1       Completed   0          6d
      ```

If it has any other state, wait a few minutes and check again (it may take some time).

7. To deploy the MicroStrategy containers, execute the following command:

   ```
   kubectl create -f <destination path>/gcxi.yaml
   ```

   where <destination path> is the folder in which the Genesys IP is stored, for example:

   ```
   kubectl create -f /genesys/gcxi/gcxi.yaml
   ```

8. Complete the following steps to verify that both GCXI pods are running:
   1. Execute the following command to verify the state of the PRIMARY pod:

      ```
      kubectl get pods | grep 'gcxi-primary*'
      ```

      The pod status should be Running, for example:

      ```
      gcxi-primary-549f6897f-zghqf      1/1       Running     0          6d
      ```

      If it has any other state, wait a few minutes and check again (it may take some time).
   2. Execute the following command to verify the state of the secondary gcxi pod:

      ```
      kubectl get pods | grep 'gcxi-secondary*'
      ```

      The pod status should be Running, for example:

      ```
      gcxi-secondary-75fdb444df-z5nbq      1/1       Running     0          6d
      ```

      If it has any other state, wait a few minutes and check again (it may take some time).

> **Important**
> The MicroStrategy server instance that runs in the container includes a temporary pre-activated key, which is required for the operation of MicroStrategy. Request a replacement key from your Genesys account representative; you need the new key to complete the procedure Install a new license key.

Configuring Ingress

By default, Kubernetes does not expose any app ports publicly. To make your app accessible, you must configure a special entity called *Ingress*. As for any Kubernetes entity, a variety of Ingress methods are supported, for more information see Kubernetes documentation. The following sections provide examples of a simple case where an NGINX daemon is run on each cluster node.

Because of differences in supported Kubernetes versions, the instructions vary depending on the release of Genesys CX Insights you have deployed (see the Product Alert for information about release support); be sure to use the instructions that are suitable for your deployment.

Procedure: Configuring Ingress in release 100.0.024 and later

Purpose: Use the steps in this example procedure to configure Ingress on selected ports, for Genesys CX Insights release 100.0.024 and later. This release requires different installation steps than used in previous releases. This procedure includes steps to remove the old controller if it was installed in a previous release.

If you are upgrading from an earlier release where you installed Ingress, you can either:

- Skip this procedure, and continue to use the old Ingress controller.
  OR

- Complete this procedure, which includes steps to remove the old controller before installing the new one.

Prerequisites

Ensure that the latest Helm version is installed on the PRIMARY machine. This procedure applies to Helm 3.7.1 or later, and NGINX Ingress controller 1.0.4 or later.

Steps

Complete the following steps on the PRIMARY machine:

1. If you previously deployed Ingress under Genesys CX Insights 9.0.012 through 100.0.023, delete the old ingress:

   ```
   kubectl delete -f <ip_path>/nginx-configmap.yaml -n ingress-nginx
   kubectl delete -f <ip_path>/nginx-daemon.yaml -n ingress-nginx
   ```

   where <ip_path> is the folder in which the previously-installed Genesys IP is stored.

2. If an error similar to `Internal error occurred: failed calling webhook...` sometimes appears, you must execute the following command to delete the legacy webhook:

   ```
   kubectl delete ValidatingWebhookConfiguration gcxi-nginx-ingress-nginx-admission
   kubectl delete namespace ingress-nginx
   ```

3. To deploy nginx-ingress controller, execute the following commands:

   ```
   helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
   helm repo add stable https://charts.helm.sh/stable
   helm repo update

   helm install ingress-nginx ingress-nginx/ingress-nginx --set
   controller.hostNetwork=true,controller.hostPort.enabled=true,controller.kind=DaemonSet,tcp.34952=genesys/
   gcxi:mstr,tcp.8180=genesys/gcxi:metrics -n ingress-nginx --create-namespace
   ```

4. Create gcxi ingress rules:

   ```
   kubectl apply -f <destination path>/gcxi-ingress.yaml
   ```

Procedure: Configuring Ingress in release 100.0.023 and earlier

Purpose: Use the steps in this example procedure to configure Ingress on selected ports. This release supports Ingress controller 0.30.0, which requires different installation steps than used in previous releases. This procedure is valid for release 9.0.012.01 through 100.0.023.

If you previously deployed Ingress in an earlier release, you can either:

- Skip this procedure, and continue to use the old Ingress controller.
  OR
- Complete this procedure, which includes steps to remove the old controller before installing the new one.

Steps

Complete the following steps on the PRIMARY machine:

1. If you previously deployed Ingress under Genesys CX Insights 9.0.011.02 or earlier, delete the old ingress:

   ```
   kubectl delete -f <destination path>/ingress.yaml
    kubectl delete -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.20.0/deploy/namespace.yaml
   ```

   where <destination_path> is the folder in which the Genesys IP (for release 9.0.011.02 or earlier) is stored.

2. Deploy ingress controller infrastructure — complete one of the following steps:
   - In **online** deployment scenarios, execute the following commands:

     ```
     kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.30.0/deploy/static/namespace.yaml
     kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.30.0/deploy/static/rbac.yaml
     ```

     OR

   - In **offline** deployment scenarios, complete the following steps:
     1. Copy the **namespace.yaml** and **rbac.yaml** files from the online machine to the offline machine, placing them in a convenient folder, such as the the folder in which the Genesys IP is stored.
     2. Execute the following commands on the offline machine:

        ```
        kubectl apply -f <path>/namespace.yaml
        kubectl apply -f <path>/rbac.yaml
        ```

        where <path> is the folder on the offline machine in which you stored the **namespace.yaml** and **rbac.yaml** files.
3. Deploy ingress controller configmaps — execute the following command to deploy ingress tcp rules:

   ```
   kubectl apply -f <destination path>/nginx-configmap.yaml
   ```

   where <destination path> is the folder in which the Genesys IP is stored.

   The **nginx-configmap.yaml** file contains the port value for non-HTTP traffic. This port is 34952 and it is used by MicroStrategy client tools, such as MicroStrategy Developer.

   For example:

   ```
   kubectl apply -f <destination path>/nginx-configmap.yaml
   ```
4. Define hostPort values:

   ```
   kubectl apply -f <destination path>/nginx-daemon.yaml
   ```

   The hostPort values specified in the **nginx-daemon.yaml** file are the ports to which your http, https, and mstr-tcp traffic will be exposed. The mstr-tcp port is used internally by

Microstrategy applications, such as Developer.

5. Create gcxi ingress rules:

```
kubectl apply -f <destination path>/gcxi-ingress.yaml
```

Configure secrets

Procedure: Configuring Kubernetes Secrets

Purpose: Use the steps in this procedure to optionally configure Kubernetes Secrets. When this feature is configured, Kubernetes stores configuration data, such as passwords, in secure Kubernetes objects. If this feature is not configured, such information is stored in plain text.

Steps

1. Open the **gcxi.properties** and **gcxi-secrets.yaml**, and review instructions and examples in the comments.
2. Configure the following variables for the secrets that are required in your environment:

```
GCXI_GIM_DB.DB_NAME: <base64 value>
 GCXI_GIM_DB.LOGIN: <base64 value>
 GCXI_GIM_DB.PASSWORD: <base64 value>
 IWD_DB.DB_NAME: <base64 value>
```

```
  IWD_DB.LOGIN: <base64 value>
  IWD_DB.PASSWORD: <base64 value>
  MSTR_PASSWORD: <base64 value>
  META_DB_ADMIN: <base64 value>
  META_DB_ADMINDB: <base64 value>
  META_DB_ADMINPWD: <base64 value>
  META_DB_PASSWORD: <base64 value>
  META_HIST_PASSWORD: <base64 value>
  TOMCAT_ADMINPWD: <base64 value>
```

Where <base64 value> is the base64-encoded values for the variable.

Note that "GCXI_GIM_DB" and "IWD_DB" have corresponding DSNDEF entries in the **gcxi.properties** file.

Ensure that the DSNDEF variables used in the secrets correspond to entries in the **gcxi.properties** file, for example:

- Secrets:

```
GCXI_GIM_DB.DB_NAME: bXlfZGI=
GCXI_GIM_DB.LOGIN: bXlfbG9naW4=
GCXI_GIM_DB.PASSWORD: bXlfcGFzc3dvcmQ=
```

- Corresponding DSNDEF entry in **gcxi.properties** (Note the presence of DB_NAME, LOGIN, PASSWORD):

```
DSNDEF1=DSN_NAME=GCXI_GIM_DB;DB_TYPE=POSTGRESQL;DB_TYPE_EX=PostgreSQL;HOST=gi2-qadb;PORT=5432;DB_NAME=;LOGIN=;PASSWORD=;
```

Note that, if any variable is present in both **gcxi-secrets.yaml** and **gcxi.properties**, the value from **gcxi-secrets.yaml** is used.

Verify the installation

Once all steps are complete, use the information in this section to verify that the installation was successful.

The installed Genesys CX Insights folder

Procedure: Verifying Genesys CX Insights installation

Purpose: Use the steps in this procedure to verify the installation. The Genesys CX Insights installation routine creates the folder shown in the figure *The installed Genesys CX Insights folder*.

Steps

After you have successfully installed Genesys CX Insights, complete the following steps:

1. Execute the following command and examine the output:

   ```
   kubectl get nodes
   ```

   The output should be similar to the following:

   ```
   NAME             STATUS    ROLES     AGE       VERSION
    spb-rhel-mstr1   Ready     primary   6d        v1.18.1
    spb-rhel-mstr2   Ready     <none>    6d        v1.18.1
   ```

2. Execute the following command and examine the output:

   ```
   kubectl get pods
   ```

   The output should be similar to the following:

   ```
   NAME                            READY     STATUS      RESTARTS   AGE
    gcxi-init-2qvcd                 0/1       Completed   0          6d
    gcxi-primary-587dc679c-fn2w4     1/1      Running     0          6d
    gcxi-postgres-77b7f946c-drck4   1/1       Running     0          6d (this line appears only if prepackaged PostgreSQL server
   is used)
    gcxi-secondary-5d9f4485bb-d8v25     1/1   Running     1          6d
   ```

3. Execute the following command and examine the output:

   ```
   kubectl get services
   ```

   The output should be similar to the following:

   ```
   NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)              AGE
    gcxi            ClusterIP   10.98.156.54    <none>        34952/TCP,8080/TCP   6d
    gcxi-postgres   NodePort    10.101.64.127   <none>        5432:31642/TCP       6d (this line appears only if prepackaged
   PostgreSQL server is used)
   ```

```
mstr-01         ClusterIP   None            <none>        34952/TCP,8080/TCP   6d
mstr-02         ClusterIP   None            <none>        34952/TCP,8080/TCP   6d
```

4. View the Genesys CX Insights reports in MicroStrategy Web to confirm that the reports are installed, by pointing your web browser to `http://<servername>:<port>/MicroStrategy/servlet/mstrWeb`, where `<servername>` is the IP or host name of any worker node, and `<port>` is the port number (usually `80`).
5. Verify the schema version.
6. Verify the Genesys CX Insights Release number.
7. View the GCXI Project in MicroStrategy Developer.

> **Important**
>
> Unlike most other Genesys applications, Genesys CX Insights is not configured as an application within Genesys Configuration Server, nor is it started (or stopped) by using the Genesys Solution Control Interface.

Procedure: Install a new License key

Purpose: Use the steps in this procedure to install a new license key. The MicroStrategy server instance that runs in the container includes a temporary pre-activated key, which is required for the operation of MicroStrategy.

Prerequisites

Obtain a new license key; contact your Genesys Customer Care representative for assistance.

Steps

1. Execute the following command to back up the GCXI meta db:

   ```
   kubectl apply -f <destination path>/gcxi-backup.yaml
   ```

   where <destination path> is the folder in which the Genesys IP is stored, for example:

   ```
   kubectl apply -f /genesys/gcxi/gcxi-backup.yaml
   ```

2. Execute the following commands to stop currently running containers:

   ```
   kubectl scale deploy/gcxi-secondary --replicas=0
   kubectl scale deploy/gcxi-primary --replicas=0
   ```

3. Edit the **gcxi.properties** file, and add the line

   ```
   MSTR_LICENSE=<your new license>
   ```

   where <your new license> is the new license key value

   This adds the MSTR_LICENSE environment variable to your Genesys CX Insights environment.

4. Execute the following commands to load gcxi.properties into Kubernetes:

   ```
   kubectl delete configmap gcxi-config
   kubectl create configmap gcxi-config --from-env-file=<destination path>/gcxi.properties --namespace genesys
   ```

   where <destination path> is the folder in which the Genesys IP is stored, for example:

```
kubectl create configmap gcxi-config --from-env-file=/genesys/gcxi/gcxi.properties --namespace genesys
```

5. Execute the following command to start the PRIMARY container:

   ```
   kubectl scale deploy/gcxi-primary --replicas=1
   ```

   Wait until PRIMARY is done (wait until Tomcat is up, and MicroStrategyWeb page is available).

6. Execute the following command to start the SECONDARY container:

   ```
   kubectl scale deploy/gcxi-secondary --replicas=1
   ```

7. Optionally, verify that the new License key is installed by checking container's log file (pod stdout). For more information, see Generating logs.

Keep in mind that you must perform additional post-installation setup steps before actively using the reports and projects. After completing the steps on this page, complete the following:

- Installing report editing software
- Post-Installation steps