Genesys Customer Experience Insights Deployment Guide

Installing Genesys CX Insights - Kubernetes using Helm

12/13/2025

Contents

Installing Genesys CX Insights - Kubernetes using Helm

Beginning with release 9.0.016, Genesys CX Insights supports deployment using Helm Charts on Kubernetes clusters. Helm Charts provide an alternative to Kubernetes descriptors. This deployment method is suitable for production environments; for other deployment options, see Choose a deployment type and Prerequisites.

This is an example scenario — This page provides a high-level outline, illustrating one scenario to help you visualize the overall process. Genesys does not provide support for Helm or other third-party products, so you must have knowledge of Helm and other products to complete this type of installation. This example is suitable for a small deployment using Kubernetes clusters on RedHat Enterprise Linux; the steps for CentOS are similar. GCXI is known to work with Helm-3, which is described on this page.

Prerequisites for deploying using Helm

Before you begin, ensure that:

- Your Kubernetes cluster is configured and running in a suitable environment, with nodes in the **Ready** state, as described in Kubernetes documentation.
- Helm-3 is installed on the Control plane node, as described in the Helm documentation.
- The images **gcxi** and **gcxi_control** are loaded and tagged on each worker node.
- On each worker node, values are set for `kernel.sem`, `vm.max_map_count`, as required by MicroStrategy. For example:

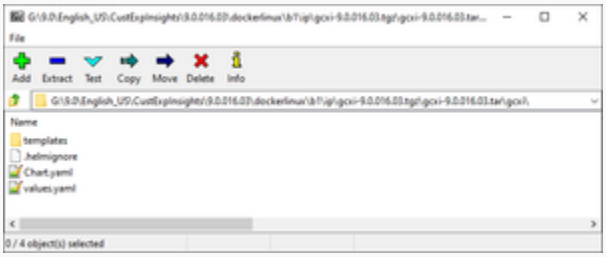
```
echo "kernel.sem = 250 1024000 250 4096" >> /etc/sysctl.conf
echo "vm.max_map_count = 5242880" >> /etc/sysctl.conf
sysctl -p
```

Deploying GCXI with Helm-3

The following procedures describe example steps to deploy GCXI with Helm. The exact steps required will vary for your environment.

Procedure: 1. Preparing for installation

Purpose: Prepare the environment, and gather files needed for deployment.



Contents of the Helm IP

Prerequisites

Within the the Genesys Customer Experience Insights package for the Docker Linux OS, look for the Helm installation package (IP) — a small TGZ file (for example **gcxi-9.0.018.00.tgz**) that contains the Helm files. You require these files to complete this procedure.

Steps

1. On the Control plane node, create a folder: **helm**.
2. Copy the Helm installation package (for example **gcxi-9.0.018.00.tgz**) into the **helm** folder, and extract the archive into a subfolder called **helm/gcxi**.
3. View the file **helm/gcxi/Chart.yaml**, and ensure that the appVersion is set to the desired GCXI version.
4. Open the file **helm/gcxi/values.yaml**, and follow the instructions it provides to guide you in creating a new file, **values-test.yaml** with appropriate settings. Save the new file in the **helm** folder.
   For example, the following content in the **values-test.yaml** file is appropriate for a simple deployment using PostgreSQL inside the container, and PersistentVolumes of the type **local**. Create appropriate content in the **values-test.yaml** file for your environment:

```
gcxi:
  env:
    GCXI_GIM_DB:
      DSNDEF:
DSN_NAME=GCXI_GIM_DB;DB_TYPE=POSTGRESQL;DB_TYPE_EX=PostgreSQL;HOST=gim_db_host;PORT=5432;DB_NAME=gi
      LOGIN: gim_login
      PASSWORD: gim_password

    IWD_DB:
      DSNDEF:
DSN_NAME=IWD_DB;DB_TYPE=POSTGRESQL;DB_TYPE_EX=PostgreSQL;HOST=iwd_db_host;PORT=5432;DB_NAME=dm_gcxi
      LOGIN: iwd_login
      PASSWORD: iwd_password

  deployment:
    deployPostgres: true
    deployLocalPV: true
    useDynamicLogPV: false
```

```
imagePullPolicy:
  worker: IfNotPresent
  control: IfNotPresent

images:
  postgres: postgres:11
```

PVCs required by GCXI

Mount Name	Mount Path (inside container)	Description	Access Type	Default Mount Point on Host (may be changed thru values) These directories MUST pre-exist on your host to accommodate the local provisioner.	Must be Shared across Nodes?	Required Node Label (applies to deafult Local PVs setup)
gcxi-backup	/genesys/gcxi_shared/backup	Backups Used by control container / jobs.	RWX	/genesys/gcxi/backup Can be overwritten by: Values.gcxi.local.pv.backup.path	Not necessarily.	gcxi/local-pv-gcxi-backup = "true"
gcxi-log	/mnt/log	MSTR logs Used by main container. The Chart allows log volumes of legacy hostPath type. This scenario is the default.	RWX	/mnt/log/gcxi subPathExpr: $(POD_NAME) Can be overwritten by: Values.gcxi.local.pv.log.path	Not necessarily.	gcxi/local-pv-gcxi-log = "true" Node label is not required if you are using hostPath volumes for logs.
gcxi-postgres	/var/lib/postgresql/data	Meta DB volume Used by Postgres container, if deployed.	RWO	/genesys/gcxi/shared Can be overwritten by: Values.gcxi.local.pv.postgres.path	Yes, unless you tie PostgreSQL container to a particular node.	gcxi/local-pv-postgres-data = "true"
gcxi-share	/genesys/gcxi_share	MSTR shared caches and cubes.	RWX	/genesys/gcxi/data subPathExpr:	Yes	gcxi/local-pv-gcxi-share =

		Used by main container.		$(POD_NAME) Can be overwritten by: Values.gcxi.local.pv.share.path		"true"

Procedure: 2. Label Nodes

Purpose: Label nodes to allocate PersistentVolumes (PVs). GCXI deployment requires three PVs, plus one for PostgreSQL deployment; see the table PVCs required by GCXI. All four PVs are linked to GCXI pods by means of PersistentVolumeClaims (PVCs). You can create PVs in advance (set **deployLocalPV: false**) or during GCXI installation (set **deployLocalPV: true**). In this example, we allow PVs to be created on all nodes.

Steps

1. Prepare nodes to keep backups — label all worker nodes and create the folder **/genesys/gcxi/backup** on each one:
    1. From the Kubernetes Control plane node, execute the following command for each worker node:

        ```
        kubectl label nodes <<worker node>> gcxi/local-pv-gcxi-backup=true
        ```

    2. On each Kubernetes worker node, execute the following command:

        ```
        mkdir /genesys/gcxi/backup/
        ```

2. Prepare nodes to keep logs — label all worker nodes and create the folder **/mnt/log/gcxi** on each one:
    1. From the Kubernetes Control plane node, execute the following command for each worker node:

        ```
        kubectl label nodes <<worker node>>  gcxi/local-pv-gcxi-log=true
        ```

    2. On each Kubernetes worker node, execute the following command:

        ```
        mkdir /mnt/log/gcxi
        ```

3. Prepare nodes to keep MicroStrategy's cache, cubes, and so on — label all worker nodes, and create the folder **/genesys/gcxi/shared**. This folder must be shared across worker nodes:
    1. From the Kubernetes Control plane node, execute the following command for each worker node:

        ```
        kubectl label nodes <<worker node>>  gcxi/local-pv-gcxi-share=true
        ```

    2. On each Kubernetes worker node, execute the following command:

        ```
        mkdir /genesys/gcxi/shared/
        ```

4. Prepare nodes to keep PostgreSQL database files — either label all worker nodes and create shared folder **/genesys/gcxi/data**, or label one node (to allow PostgreSQL to run only on it) and create the folder on it:
    1. From the Kubernetes Control plane node, execute the following command for the worker node where PostgreSQL will run:

        ```
        kubectl label nodes <<worker node>>  gcxi/local-pv-postgres-data=true
        ```

    2. On the Kubernetes worker node where PostgreSQL will run, execute the following command:

        ```
        mkdir /genesys/gcxi/data
        ```

Procedure: 3. Deploying GCXI

Purpose: Deploy GCXI. This procedure provides steps for environments without LDAP — for environments that include LDAP (or other features not supported in **values.yaml**) you can pass container environment variables such MSTR_WEB_LDAP_ON=true using the **gcxi.envvars** file (for example: `--set-file gcxi.envext=gcxi.envvars`).

Steps

1. For debug purposes, execute the following commands to render templates without installing:

    ```
    helm template --debug -f values-test.yaml gcxi-helm gcxi/
    ```

    Kubernetes descriptors are displayed. The values you see are generated from Helm templates, and based on settings from **values.yaml** and **values-test.yaml**. Ensure that no errors are displayed; you will later apply this configuration to your Kubernetes cluster.

2. To deploy GCXI, execute the following command:

    ```
    helm install --debug --namespace gcxi --create-namespace -f values-test.yaml
    gcxi-helm gcxi/
    ```

    This process takes several minutes. Wait until all objects are created and allocated, and the Kubernetes descriptors applied to the environment appear.

    Genesys recommends that you avoid using helm install with the `--wait` option when deploying GCXI. If you use `--wait`, the Helm architecture post-install hook (which in this case is the gcxi_init job) won't be triggered properly. For more information, see the Helm documentation.

3. To check the installed Helm release, execute the following command:

    ```
    helm list --all-namespaces
    ```

or alternatively, execute the following command:

```
helm list -n gcxi
```

4. To check GCXI Kubernetes objects created by Helm in gcxi release, execute the following command:

```
kubectl get all -n gcxi
```

Procedure: 4. Configuring Ingress

Purpose: To make GCXI accessible from outside the cluster, a special entity called Ingress must be configured. You can find detailed instructions to deploy NGINX Ingress Controller here. The following steps provide an example only.

Steps

1. Log in to the control plane node.
2. Execute the following commands to add the Helm repository:

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

3. Execute the following commands to install the NGINX chart:

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

4. Execute the following commands to verify that the NGINX chart is successfully installed:

```
helm list -n ingress-nginx
```

5. Execute the following commands to verify that NGINX pods are successfully deployed on all cluster nodes:

```
kubectl get pod -n ingress-nginx
```

Maintenance Procedures

This section provides additional procedures, such as troubleshooting steps.

Procedure: Troubleshooting

Purpose: Use the instructions in this section only if you encounter errors or other difficulties. Problems with the deployment are most often associated with the following three kinds of objects:

- PVs
- PVCs
- pods

Steps

1. To list the objects that might cause problems, execute the following commands:

   ```
   kubectl get pv -o wide

   kubectl get pvc -n gcxi -o wide

   kubectl get po -n gcxi -o wide
   ```

2. Examine the output from each **get** command.
3. If any of the objects have a non-ready state (for example, **Unbound** (PVCs only), **Pending**, or **CrashLoop**) execute the following command to inspect the object more closely using **describe**:

   ```
   kubectl describe <type> <name>
   ```

   For example:

   ```
   kubectl describe po gcxi-0
   ```

4. In the **describe** output, inspect the section **Events**.

Procedure: Upgrade GCXI using Helm

Purpose: In scenarios where you have previously deployed Genesys CX Insights using Helm, use the instructions in this procedure to upgrade to a newer Genesys CX Insights release.

Prerequisites

Obtain the Helm installation package (IP) before you begin. The Helm IP has a file name in the

format gcxi-<release>.tgz, for example **gcxi-018.00.tgz**. If SAML is enabled in your environment, before you update GCXI, see the instructions in the *Genesys CX Insights User's Guide* pertaining to updating Genesys CX Insights when SAML is enabled.

Steps

1. On the Control Plane node, set the current directory to the helm folder, and execute the following command to create a subfolder to differentiate this release from others:

   ```
   mkdir helm_<folder>
   ```

   where <folder> is the folder in which to extract the Helm package. For example **018.00**.

2. Execute the following command to extract the Helm package:

   ```
   tar xvzf <helm IP file name> -C helm_<folder>
   ```

   where <folder> is the folder you created in the previous step, and <helm IP file name> is the name of the gcxi Helm package you are installing.

   For example:

   ```
   mkdir helm_018.00; tar xvzf gcxi-9.0.018.00.tgz -C helm_018.00
   ```

3. Execute the following command to remove Ingress:

   ```
   helm delete gcxi-nginx -n ingress-nginx
   ```

4. From the folder where you deployed the previous release of GCXI using Helm, copy the file **values-test.yaml** into the new folder.

5. Navigate to the new folder (for example, using the CD command), and execute the following command:

   ```
   helm upgrade --debug gcxi-helm gcxi/ --namespace gcxi --create-namespace -f
   values-test.yaml
   ```

6. Execute the following command to configure Ingress:

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

Procedure: Uninstall GCXI

Purpose: To remove GCXI

Steps

1. Execute the following command to remove Ingress:

   ```
   helm delete gcxi-nginx -n ingress-nginx
   ```

2. Execute the following command to remove GCXI:

   ```
   helm uninstall gcxi-helm -n gcxi
   ```

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

- Installing report editing software
- Post-Installation steps