Genesys Pulse Private Edition Guide

Upgrade, roll back, or uninstall Genesys Pulse

5/12/2026

Contents

- 1 Supported upgrade strategies
- 2 Timing
  - 2.1 Scheduling considerations
- 3 Monitoring
- 4 Preparatory steps
- 5 Rolling Update
  - 5.1 Rolling Update: Upgrade
  - 5.2 Rolling Update: Verify the upgrade
  - 5.3 Rolling Update: Rollback
  - 5.4 Rolling Update: Verify the rollback
- 6 Uninstall

Learn how to upgrade, roll back, or uninstall Genesys Pulse.

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
>
> The instructions on this page assume you have deployed the services in service-specific 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.

Supported upgrade strategies

Genesys Pulse supports the following upgrade strategies:

Service	Upgrade Strategy	Notes
Genesys Pulse	Rolling Update	

For a conceptual overview of the upgrade strategies, refer to Upgrade strategies in the Setting up Genesys Multicloud CX Private Edition guide.

Timing

A regular upgrade schedule is necessary to fit within the Genesys policy of supporting N-2 releases, but a particular release might warrant an earlier upgrade (for example, because of a critical security fix).

If the service you are upgrading requires a later version of any third-party services, upgrade the third-party service(s) before you upgrade the private edition service. For the latest supported versions of third-party services, see the Software requirements page in the suite-level guide.

Scheduling considerations

Genesys recommends that you upgrade the services methodically and sequentially: Complete the upgrade for one service and verify that it upgraded successfully before proceeding to upgrade the next service. If necessary, roll back the upgrade and verify successful rollback.

Monitoring

Monitor the upgrade process using standard Kubernetes and Helm metrics, as well as service-specific metrics that can identify failure or successful completion of the upgrade (see Observability in Genesys Pulse).

Genesys recommends that you create custom alerts for key indicators of failure — for example, an alert that a pod is in pending state for longer than a timeout suitable for your environment. Consider including an alert for the absence of metrics, which is a situation that can occur if the Docker image is not available. Note that Genesys does not provide support for custom alerts that you create in your environment.

Preparatory steps

Ensure that your processes have been set up to enable easy rollback in case an upgrade leads to compatibility or other issues.

Each time you upgrade a service:

1. Review the release note to identify changes.
2. Ensure that the new package is available for you to deploy in your environment.
3. Ensure that your existing **-values.yaml** file is available and update it if required to implement changes.

Use the following commands to get the updated Helm charts in your repository:

```
helm repo update
helm search repo pulsehelmrepo/
```

where is the name of the Helm chart you need. To update shared and tenant provisioning, run the command for each of the following charts:

- init
- pulse
- init-tenant
- dcu
- lds
- permissions

For example, for the **pulse** Helm chart:

```
helm repo update
helm search repo pulsehelmrepo/pulse
```

Rolling Update

Rolling Update: Upgrade

Execute the following command to upgrade :

```
helm upgrade --install  -f -values.yaml  -n
```

Tip: If your review of Helm chart changes (see Preparatory Step 3) identifies that the only update you need to make to your existing **-values.yaml** file is to update the image version, you can pass the image tag as an argument by using the `--set` flag in the command:

```
helm upgrade --install  -f -values.yaml  --set .image.tag=
```

You can find additional information about the `helm upgrade --install` commands on the Shared Provisioning and Tenant Provisioning pages.

The following sections describe recommended arguments to use when you run the Helm upgrade command:

Shared provisioning

When you run the Helm upgrade command for Shared provisioning:

- For the **init** Helm chart, set the **wait** and **wait-for-jobs** flags, so that Helm waits until all jobs have completed before reporting success:

  ```
  helm upgrade --install pulse-init pulsehelmrepo/init --wait --wait-for-jobs --
   version= --namespace=pulse -f values-override-init.yaml
  ```

  This command also updates the schema, if an update is required.

- For the **pulse** Helm chart, set the **wait** flag, so that Helm waits until all pods and other Kubernetes resources in the replica set are in Ready state before reporting success:

  ```
  helm upgrade --install pulse pulsehelmrepo/pulse --wait --version=  --namespace=pulse
   -f values-override-pulse.yaml
  ```

If installation is successful, the command finishes with exit code `0`.

Tenant provisioning

When you run the Helm upgrade command for Shared provisioning:

- For the **init-tenant** Helm chart, set the **wait** and **wait-for-jobs** flags, so that Helm waits until all jobs have completed before reporting success:

```
helm upgrade --install "pulse-init-tenant-" pulsehelmrepo/init-tenant --wait --wait-
 for-jobs --version=""--namespace=pulse -f values-override-init-tenant.yaml
```

- For the **dcu**, **lds**, **lds-vq**, and **permissions** Helm charts, set the **wait** flag, so that Helm waits until all pods and other StatefulSet Kubernetes resources are in Ready state before reporting success:

```
helm upgrade --install "pulse-lds-" pulsehelmrepo/lds --wait --version= --
 namespace=pulse -f values-override-lds.yaml
 helm upgrade --install "pulse-lds-vq-" pulsehelmrepo/lds --wait --version= --
 namespace=pulse -f values-override-lds.yaml -f values-override-lds-vq.yaml

helm upgrade --install "pulse-permissions-" pulsehelmrepo/permissions --wait --
 version="" --namespace=pulse -f values-override-permissions.yaml
```

If installation is successful, the command finishes with exit code `0`.

You can optionally omit the argument `--version=""` to upgrade to the latest available version.

Rolling Update: Verify the upgrade

Follow usual Kubernetes best practices to verify that the new service version is deployed. See the information about initial deployment for additional functional validation that the service has upgraded successfully.

Rolling Update: Rollback

Execute the following command to roll back the upgrade to the previous version:

```
helm rollback
```

or, to roll back to an even earlier version:

```
helm rollback
```

Alternatively, you can re-install the previous package:

1. Revert the image version in the `.image.tag` parameter in the **-values.yaml** file. If applicable, also revert any configuration changes you implemented for the new release.
2. Execute the following command to roll back the upgrade:

   ```
   helm upgrade --install  -f -values.yaml
   ```

   Tip: You can also directly pass the image tag as an argument by using the `--set` flag in the command:

   ```
   helm upgrade --install  -f -values.yaml  --set .image.tag=
   ```

Rolling Update: Verify the rollback

Verify the rollback in the same way that you verified the upgrade (see Rolling Update: Verify the upgrade).

Uninstall

> **Warning**
> Uninstalling a service removes all Kubernetes resources associated with that service. Genesys recommends that you contact Genesys Customer Care before uninstalling any private edition services, particularly in a production environment, to ensure that you understand the implications and to prevent unintended consequences arising from, say, unrecognized dependencies or purged data.

Execute the following command to uninstall :

```
helm uninstall  -n
```

For example, to uninstall:

- init:

  ```
  helm uninstall pulse-init --namespace=pulse
  ```

- pulse:

  ```
  helm uninstall pulse --namespace=pulse
  ```

- init-tenant:

  ```
  helm uninstall pulse-init-tenant- --namespace=pulse
  ```

- dcu:

  ```
  helm uninstall pulse-dcu- --namespace=pulse
  ```

- lds:

  ```
  helm uninstall pulse-lds- --namespace=pulse
   helm uninstall pulse-lds-vq- --namespace=pulse
  ```

- permissions:

  ```
  helm uninstall pulse-permissions- --namespace=pulse
  ```

> **Important**
> If you want to remove all custom Pulse dashboard configurations (shared or personal) and all widget templates, you must manually delete the Pulse database schema, which also deletes all data in the schema.