Designer Private Edition Guide

Upgrade, roll back, or uninstall Designer

5/14/2026

Contents

Learn how to upgrade, roll back, or uninstall Designer.

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

Designer supports the following upgrade strategies:

Service	Upgrade Strategy	Notes
Designer	• Rolling Update • Blue/Green • Canary	Canary is used in combination with Blue/Green.
Designer Application Server	• Rolling Update • Blue/Green • Canary	Canary is used in combination with Blue/Green.

The upgrade or rollback process to follow depends on how you deployed the service initially. Based on the deployment strategy adopted during initial deployment, refer to the corresponding upgrade or rollback section on this page for related instructions.

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 Designer).

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.

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=
```

Follow the same instructions to upgrade Designer and DAS. For example, the respective commands are:

- Designer:

  ```
  helm upgrade --install --namespace designer designer -f designer-values.yaml
   designer-100.0.112+1401.tgz  --set designer.image.tag=100.0.112.11
  ```

- DAS:

  ```
  helm upgrade --install --namespace designer designer-das -f designer-das-values.yaml
   designer-das-100.0.112+1401.tgz  --set das.image.tag=9.0.111.05.5
  ```

If you are using the --set flag in the helm upgrade command to populate the `designer.designerConfig.envs` or `das.dasConfig.envs` values, use `--set-string`, for example:

- Designer: `--set-string designer.designerConfig.envs.DES_ES_PORT="8080"`
- DAS: `--set-string das.dasConfig.envs.DAS_SERVICES_ELASTICSEARCH_PORT="9200"`

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=
   ```

Follow the same instructions to roll back Designer and DAS. For example, the respective commands are:

- Designer:

  ```
  helm upgrade --install --namespace designer designer -f designer-values.yaml
   designer-100.0.112+1401.tgz  --set designer.image.tag=100.0.112.11
  ```

- DAS:

  ```
  helm upgrade --install --namespace designer designer-das -f designer-das-values.yaml
   designer-das-100.0.112+1401.tgz  --set das.image.tag=9.0.111.05.5
  ```

Rolling Update: Verify the rollback

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

- Ensure that the image version in the -values.yaml file reflects the version that you rolled back to.

Blue/Green

Blue/Green: Upgrade Designer

1. Identify the current production color by checking the Designer ingress rules:
   `kubectl describe ingress designer-ingress`

   Green is the production color in the below example as the production host name points to the green service.

   kubectl describe ingress designer-ingress

   ```
   Host                        Path  Backends
   ----                        ----  --------
   designer.example.com        /   designer-green:http (10.244.0.23:8888)
   designer.green.example.com  /   designer-green:http (10.244.0.23:8888)
   designer.blue.example.com   /   designer-blue:http (10.244.0.45:8888)
   ```

2. Deploy the Designer service on to the non-production color (in this example, blue is the non-production color and assuming the service name is designer-blue):

```
helm upgrade --install --namespace designer designer-blue -f designer-values.yaml
designer-100.0.112+1401.tgz --set designer.deployment.strategy=blue-green --set
designer.image.tag=100.0.111.05.5 --set designer.deployment.color=blue
```

Use the non-production host name to access the non-production color. For example, `designer.blue.example.com`). You can use this URL for testing.

NodePort Service

The `designer-green` release creates a service called `designer-green` and the `designer-blue` release creates a service called `designer-blue`. If you are using NodePort services, ensure that the value of `designer.service.nodePort` is not the same for both the releases. In other words, you should assign dedicated node ports for the releases. The default value for `designer.service.nodePort` is **30180**. If this was applied to `designer-green`, use a different value for `designer-blue`, for example, **30181**. Use the below helm command to achieve this:

```
helm upgrade --install --namespace designer designer-blue -f designer-values.yaml
designer-100.0.112+1401.tgz --set designer.deployment.strategy=blue-green --set
designer.image.tag=100.0.111.05.5 --set designer.deployment.color=blue --set
designer.service.nodePort=30181
```

Cutover

Once testing is completed on the non-production color, move traffic to the new version by updating the Ingress rules:

- Update the Designer Ingress with the new deployment color by running the following command (in this case, blue is the new deployment color, that is, the non-production color):

```
helm upgrade --install --namespace designer designer-ingress -f designer-values.yaml
designer-100.0.112+1401.tgz --set designer.deployment.strategy=blue-green-ingress --
set designer.deployment.color=blue
```

Verify the upgrade

- Verify the ingress rules by running the following command:
  `kubectl describe ingress designer-ingress`
  The production host name must point to the new color service, that is, `blue`.

Blue/Green: Rollback Designer

To roll back the upgrade, modify the ingress rules to point back to the old deployment pods (green, in this example) by performing a cutover again.

- Perform a cutover using the following command:

```
helm upgrade --install --namespace designer designer-ingress -f designer-values.yaml
designer-100.0.112+1401.tgz --set designer.deployment.strategy=blue-green-ingress --
set designer.deployment.color=green
```

Verify the rollback

- Verify the rollback in the same way that you verified the upgrade (see Blue-Green: Verify the upgrade). The type label must have the active color's label, that is, `color=green`.

Blue/Green: Upgrade DAS

1. Identify the current production color by checking the `designer-das` service selector labels:

   ```
   kubectl describe service designer-das
   ```

   Green is the production color in the below example as the selector label is `color=green`.

   kubectl describe service designer-das

   ```
   Selector:                 color=green
   ```

2. Deploy the DAS service on to the non-production color (in this example, blue is the non-production color and assuming the service name is `designer-das-blue`):

   ```
   helm upgrade --install --namespace designer designer-das-blue -f das-values.yaml
   designer-das-100.0.106+1401.tgz --set das.deployment.strategy=blue-green --set
   das.image.tag=9.0.111.05.5 --set das.deployment.color=blue
   ```

Use the non-production service name to access the non-production color.

NodePort Service

The `designer-das-green` release creates a service called `designer-das-green` and the `designer-das-blue` release creates a service called `designer-das-blue`. If you are using NodePort services, ensure that the value of `designer.service.nodePort` is not the same for both the releases. In other words, you should assign dedicated node ports for the releases. The default value for `designer.service.nodePort` is **30280**. If this was applied to `designer-das-green`, use a different value for `designer-das-blue`, for example, **30281**. Use the below helm command to achieve this:

```
helm upgrade --install --namespace designer designer-das designer-das-100.0.106+xxx.tgz -f
designer-das-values.yaml --set das.deployment.strategy=blue-green-service --set
das.deployment.color=green --set das.service.nodePort=30281
```

Cutover

Once testing is completed on the non-production color, move traffic to the new version by updating the `designer-das` service.

- Update the `designer-das` service with the new deployment color (in this example, blue is the new deployment color, that is, non-production color)

  ```
  helm upgrade --install --namespace designer designer-das-service -f designer-das-
  values.yaml designer-das-100.0.106+1401.tgz --set das.deployment.strategy=blue-green-
  service --set das.deployment.color=blue
  ```

Verify the upgrade

- Verify the service by executing the `kubectl describe service designer-das` command.
  The type label must have the active color's label, that is, `color=blue`.

Blue/Green: Rollback DAS

To roll back the upgrade, perform a cutover again to point the service back to the old deployment (green).

- Perform a cutover using the following command:

```
helm upgrade --install --namespace designer designer-das-service -f designer-das-
values.yaml designer-das-100.0.106+1401.tgz --set das.deployment.strategy=blue-green-
service --set das.deployment.color=green
```

Verify the rollback

- Verify the rollback in the same way that you verified the upgrade (see Blue-Green: Verify the upgrade).
  The type label must have the active color's label, `color=green`.

Canary

Canary is optional and is only used along with Blue-Green. It is recommended in production. Canary pods are generally used to test new versions of images with live traffic. You will not use Canary pods when you are installing the Designer and DAS services for the first time. You will only use Canary pods for testing the new versions when upgrading the services after initial deployment.

1. Identify the current production color by checking the `designer-das` service selector labels (`kubectl describe service designer-das`). Green is the production color in the below example as the selector label is `color=green`.

   kubectl describe service designer-das

   ```
   Selector:                 color=green
   ```

2. To deploy canary pods, the `das.deployment.strategy` value must be set to `canary` in the **designer-das-values.yaml** file or using the `--set` flag as shown in the command below:
   `helm upgrade --install --namespace designer designer-das-canary -f das-values.yaml designer-das-100.0.106+xxxx.tgz --set das.deployment.strategy=canary --set das.image.tag=9.0.1xx.xx.xx --set das.deployment.color=green`
   The values.yaml overrides passed as an argument to the above Helm upgrade command:
   `das.deployment.strategy=canary` - This denotes that the Helm install will create canary pods.
   `das.deployment.color=green` - This denotes that the current production (active) color is green.

> **Important**
>
> To make sure Canary pods receive live traffic, they have to be exposed to the `designer-das` service by setting `das.deployment.color=`, which is obtained from step 1.

3. Once canary pods are up and running, ensure that the designer-das service points to the canary pods using the `kubectl describe svc designer-das` command.

```
Endpoints: 10.206.0.101:8081,10.206.0.162:8081,10.206.0.90:8081
```

   The IP address present in the Endpoints must match the IP address of the canary pod. The canary pod's IP address is obtained using the `kubectl describe pod` command.

```
IP: 10.206.0.90
IPs:
  IP: 10.206.0.90
```

Cleaning up

After completing canary testing, the canary pods must be cleaned up. The `das.deployment.replicaCount` must be made zero and the release is upgraded. It can be changed in the **designer-das-values.yaml** file or through the `--set` flag as follows:

- `helm upgrade --install --namespace designer designer-das-canary -f das-values.yaml designer-das-100.0.106+xxxx.tgz --set das.deployment.strategy=canary --set das.image.tag=9.0.1xx.xx.xx --set das.deployment.color=blue --set das.deployment.replicaCount=0`

Post-upgrade procedures

Upgrading the Designer workspace

Workspace resources must be upgraded after cutover. Perform the following steps to upgrade the system resources in the Designer workspace:

1. Log in to one of the Designer pods using the `kubectl exec -it` bash command.
2. Execute the following migration command (this creates new directories/new files introduced in the new version):
   `node ./bin/cli.js workspace-upgrade -m -t`
3. Execute the workspace resource upgrade command (this upgrades system resources, such as system service PHP files, internal audio files and callback resources):
   `node ./bin/cli.js workspace-upgrade -t`
   In the above command, `contact_center_id` , is the Contact Center ID created in GWS for this tenant (workspace resources are located under the Contact Center ID folder (*/workspaces//workspace*)).

Elasticsearch maintenance recommendations

To help you better manage your indexes and snapshots, and to prevent too many indexes from creating an overflow of shards, Genesys recommends that you set up a scheduled execution of Elasticsearch Curator with the following two actions:

- Delete indexes older than the given threshold according to the index name and mask.
  - `sdr-*` (3 months)
  - `audit-*` (12 months)
- Make a snapshot of each index:
  - `sdr-*` (yesterday and older)
  - `audit-*`
  - `kibana-int-*`

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
```