This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Web Services and Applications Private Edition Guide

Deploy GWS Services

4/24/2024

Contents

- 1 Assumptions
- 2 Prepare your environment
  - 2.1 GKE
  - 2.2 AKS
- 3 Deploy
- 4 Validate the deployment
- 5 Next steps

Learn how to deploy GWS Services 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 for the full list of prerequisites required to deploy Genesys Web Services and Applications.

Prepare your environment

To prepare your environment for the deployment, complete the steps in this section for Google Kubernetes Engine (GKE) or Azure Kubernetes Service (AKS).

GKE

Log in to the GKE cluster from the host where you will run the deployment:

```
gcloud container clusters get-credentials
```

Create a new namespace for Genesys Web Services and Applications with a JSON file that specifies the namespace metadata. For example, **create-gws-namespace.json**:

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

Execute the following command to create the namespace:

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

Confirm the namespace was created:

```
kubectl describe namespace gws
```

AKS

Log in to the AKS cluster from the host where you will run the deployment:

```
az aks get-credentials --resource-group  --name  --admin
```

Create a new namespace for Genesys Web Services and Applications with a JSON file that specifies the namespace metadata. For example, **create-gws-namespace.json**:

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

Execute the following command to create the namespace:

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

Confirm the namespace was created:

```
kubectl describe namespace gws
```

Deploy

To deploy GWS Services, you'll need the Helm package and your override files. Copy **values.yaml**, **versions.yaml** and the Helm package (**gws-services-.tgz**) to the installation location. For debugging purposes, use the following command to render templates without installing so you can check that resources are created properly:

```
helm template --debug /gws-services-.tgz -f values.yaml -f versions.yaml
```

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

Now you're ready to deploy GWS Services:

> **Important**
>
> If you have configured TLS for connections to third-party services or legacy Genesys servers, make sure to point to your local files in the `helm upgrade` command.

```
helm upgrade --install gws-services /gws-services --version= -n gws -f ./override.gws-
services.values.yaml -f ./versions.yaml
```

Validate the deployment

First check the installed Helm release:

```
helm list –n gws
```

The result should show the **gws-services** deployment details. For example:

```
NAME            NAMESPACE   REVISION    UPDATED
STATUS      CHART                   APP VERSION
gws-services    gws         1           2021-05-19  11:49:49.2243107 +0530 +0530
deployed    gws-services-1.0.18     1.0
```

Check the **gws-services** status:

```
helm status gws-services
```

The result should show the namespace details with a status of deployed:

```
NAME: gws-services
LAST DEPLOYED: Wed May 19 11:49:49 2021
NAMESPACE: gws
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Check the GWS Kubernetes objects created by Helm:

```
kubectl get all -n gws
```

The result should show all the created pods, services, ConfigMaps, and so on.

Finally, confirm Agent Setup is accessible by navigating to **gws./ui/provisioning** in a web browser.

Next steps

- Configure GWS Ingress
- Deploy GWS Ingress
- Provision Genesys Web Services and Applications