Genesys Authentication Private Edition Guide

Deploy Genesys Authentication

7/12/2025

Contents

Learn how to deploy Genesys Authentication 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 Authentication.

Prepare your environment

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

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 Authentication with a JSON file that specifies the namespace metadata. For example, **create-gauth-namespace.json**:

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

Execute the following command to create the namespace:

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

Confirm the namespace was created:

```
kubectl describe namespace gauth
```

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 Authentication with a JSON file that specifies the namespace metadata. For example, **create-gauth-namespace.json**:

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

Execute the following command to create the namespace:

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

Confirm the namespace was created:

```
kubectl describe namespace gauth
```

Deploy

To deploy Genesys Authentication, you'll need the Helm package and your overrides file. Copy **values.yaml** and the Helm package (**gauth-.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 /gauth-.tgz -f values.yaml
```

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

Now you're ready to deploy Genesys Authentication:

```
helm install gauth ./gauth-.tgz -f values.yaml -n gauth
```

Configure external access

Follow the instructions for either GKE or AKS to make the Genesys Authentication services accessible from outside the cluster.

Provision ingresses for GKE or AKS

After deploying, make Genesys Authentication services accessible from outside the GKE or AKS cluster using the NGINX Ingress Controller.

Create a YAML file called **gauth-ingress.yaml** with the content below. **Note:** Replace **gws.** and **gauth.** with your GWS and Genesys Authentication domains, such as `gws.test.dev`.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gauth-gws-ingress
  namespace: gauth
  annotations:
    # add an annotation indicating the issuer to use.
    cert-manager.io/cluster-issuer: "selfsigned-cluster-issuer"
    # Custom annotations for NGINX Ingress Controller
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/use-regex: "true"
spec:
  rules:
  - host: gws. - e.g. gws.test.dev
    http:
      paths:
        - path: /ui/auth/.*
          backend:
            serviceName:  gauth-auth-ui
            servicePort: 80
        - path: /auth/.*
```

```
          backend:
            serviceName:  gauth-auth
            servicePort: 80
        - path: /environment/.*
          backend:
            serviceName:  gauth-environment
            servicePort: 80
  tls:
  - hosts:
    - gws. - e.g. gws.test.dev
    secretName: gauth-gws-ingress-cert
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gauth-gauth-ingress
  namespace: gauth
  annotations:
    # add an annotation indicating the issuer to use.
    cert-manager.io/cluster-issuer: "selfsigned-cluster-issuer"
    # Custom annotations for NGINX Ingress Controller
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/use-regex: "true"
spec:
  rules:
  - host: gauth. - e.g. gauth.test.dev
    http:
      paths:
        - path: /ui/auth/.*
          backend:
            serviceName:  gauth-auth-ui
            servicePort: 80
        - path: /auth/.*
          backend:
            serviceName:  gauth-auth
            servicePort: 80

        - path: /environment/.*
          backend:
            serviceName:  gauth-environment
            servicePort: 80
  tls:
  - hosts:
    - gauth. - e.g. gauth.test.dev
    secretName: gauth-gauth-ingress-cert
```

Create ingresses with the following command:

```
kubectl apply -f gauth-ingress.yaml -n gws
```

Validate the deployment

Check the installed Helm release:

```
helm list
```

The results should show the Genesys Authentication deployment details. For example:

```
NAME    NAMESPACE       REVISION        UPDATED
STATUS          CHART           APP VERSION
gauth   gauth           1               2021-05-20 11:56:32.5531685 +0530 +0530
deployed        gauth-0.1.77    0.1
```

Check the **gauth** namespace status:

```
helm status gauth
```

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

```
NAME: gauth
LAST DEPLOYED: Thu May 20 11:56:32 2021
NAMESPACE: gauth
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Check the Genesys Authentication Kubernetes objects created by Helm:

```
kubectl get all -n gauth
```

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

Finally, verify that you can now access Genesys Authentication at the following URL: https:///ui/auth/sign-in.html