Genesys Callback Private Edition Guide

Deploy Genesys Engagement Service

4/14/2024

Contents

Learn how to deploy Genesys Engagement Service 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.

For information about downloading containers, see Downloading your Genesys Multicloud CX containers.

> **Important**
> Before deploying Genesys Engagement Service, review the Before you begin page for the full list of prerequisites.

Before you begin

Before you begin deployment, collect the following information – it is used during deployment:

- External GWS URL
- Internal GWS URL

- GES Redis host
- ORS Redis host
- GES internal URL
- GES external URL
- Postgres host

Deploy the service

1. Because Genesys Engagement Service (GES) relies on other services, you must ensure that the following have been deployed and configured successfully before you deploy GES:
   - Voice Microservices
   - Designer
   - GWS
   - Agent Setup (you require Agent Setup to create user logins for GES and to assign the necessary privileges)
2. Create dedicated Redis and Postgres instances for GES. Note the hostname and the credentials to authenticate and then provision Kubernetes secrets to store the following:
   - REDIS-CACHEKEY
   - DB-USER
   - DB-PASSWORD
3. Create the GES project.
4. Provision a set of GWS client credentials for GES.

   You need the Tenant ID for this step. You can use a `curl` command such as the following sample to find the Tenant ID:

   ```
   curl --location --request GET 'https://gauth-int./environment/v3/environments' \
    --header 'Content-Type: application/json' \
    --header 'Authorization: Basic ' \
   ```

   Create the GWS key:

   ```
   curl -X "POST" "http:///auth/v3/ops/clients" \
        -H 'Content-Type: application/json; charset=utf-8' \
        -u 'opsAdmin:opsPass' \
        -d $'{
     "data": {
       "authorities": [
         "ROLE_INTERNAL_CLIENT"
       ],
       "contactCenterIds": [
         ""
       ],
   ```

```
      "redirectURIs": [
        "https://ges-external-ges.apps./ges/ui/login.html",
        "http://ges-internal-ges.apps./ges/ui/login.html"
      ],
      "scope": [
        "*"
      ],
      "authorizedGrantTypes": [
        "client_credentials",
        "authorization_code",
        "refresh_token",
        "implicit",
        "password"
      ],
      "refreshTokenExpirationTimeout": 43200,
      "accessTokenExpirationTimeout": 43200,
      "clientType": "CONFIDENTIAL",
      "client_secret": "",  # NOTE: This is your unencrypted GWS client
  secret.Encrypted it will be your AUTHENTICATION-CLIENT-SECRET
      "name": "ges_client",                      # It is possible to change this name.
      "client_id": "ges_client"                  # It is possible to change this
  name. This is what is stored in AUTHENTICATION-CLIENT-ID
    }
  }'
```

In the preceding configuration, add values specific to your environment:

- : Specify the internal API of GWS.
- `contactCenterIds`: Specify your Tenant ID.
- `redirectURIs`: Specify the URL(s) that Designer uses.
- : Specify the domain address for the environment.
- `-u 'ops:ops'`: This is the default delivered for Tenants.
- `client_secret`: Specify any value (used in the secret).
- `name`: Specify any value.
- `client_id`: Specify your client ID (used in the secret).

5. Provision secrets for GES.
   - Create an infrastructure secret. This secret contains the username (DB), password (DB), and REDIS_CACHEKEY (GES Redis password; this is optional). Execute the following command:

     ```
     kubectl create secret generic ges-secrets-infra --from-literal=DB-USER= --from-
     literal=DB-PASSWORD= -- from-literal=REDIS_CACHEKEY=
     ```

   - Create an ORS Redis secret. This secret contains connection details for the Redis ORS stream. This has the servername (Redis Voice Cluster Hostname – redis-ors-stream), port (Redis Voice Cluster Port), and password (Redis Voice Cluster Password – if used). Execute the following command:

     ```
     kubectl create secret generic voice-redis-ors-stream --from-literal=voice-redis-
     ors-stream='{"password":"", "port":"", "rejectUnauthorized":"true",
     "servername":""}'
     ```

   - Create the GWS client secret. This secret contains the details for your GWS client (created earlier). Execute the following command:

     ```
     kubectl create secret generic gws-client-credentials --from-
     literal=AUTHENTICATION-CLIENT-ID= --from-literal=AUTHENTICATION-CLIENT-SECRET=
     ```

For more information about how GES leverages secrets and how they can be provisioned, see secrets and Using secrets to integrate with other software.

6. Download the required Helm chart release from the JFrog repository. For information about how to download the Helm charts, see Downloading your Genesys Multicloud CX containers. See Helm charts and containers for Callback for the Helm chart version you must download for your release.
7. Install the GES Helm charts:

```
helm install ges ./ -f ges-values.yaml
```

8. Using information in the Override Helm chart values section, create a Helm values file that describes the details of your GES deployment, secrets, and environment. A base Helm **values.yaml** file is included with the GES Helm charts that you receive from Genesys. The GES base file uses the filename, **ges-base-values.yaml**.
   Be sure to specify the following values in the **ges-values.yaml** overrides file for your GES deployment:

   - : Specify the data center region (Consul).
   - : Specify the address for the GES Redis server.
   - : Specify the port for the GES Redis server.
   - : Specify the address for the GES Postgres server.
   - : Specify the database name for GES Redis.
   - : Specify the internal cluster address for the gauth Authentication server.
   - : Specify the internal cluster address for the gauth Environment server.
   - : Specify the internal address for the GWS Service.
   - : Specify the external gauth address.
   - : Specify the address for the Redis-ORS-stream Redis server.
   - : Specify the port for the Redis-ORS-stream Redis server.
   - : Specify your "pull" secret name.

   This is a sample **ges-values.yaml** file:

```
ges:

  affinity: {}

  labels:
    globalLabels:
      version: "{{ .Release.Name }}"
    podLabels: {}
    serviceLabels: {}
  annotations:
    globalAnnotations: {}
    podAnnotations: {}

  configMap:
    env:
      extendEnv: "false"
      regionAffinity: ""
      devOpsAccessGroups: "Platform Read Only,Platform Provisioning,Platform
Internal Administrator"
```

```
      log:
        level: "DEBUG"
        max_len: 2048

      server:
        internal_port: 3050
        external_port: 3005
        internal_url: "ges.ges"

      ui:
        devops_username: "admin"
        devops_password: "letmein"


    integrations:

      # Omit if using secrets to supply Redis information
      redis:
        host:
        port:
        secure: "false"

      # Omit all but Secure if using secrets to supply DB information
      db:
        host:
        name:
        secure: "false"

      gws:
        auth:
        env:
        conf:
        public_auth:


      vmcs:
        redis_host:
        redis_port:
        redis_cluster_mode: "false"
        redis_stream_name: "NewCallbackStream"
        ors_redis_location: "{{ .Values.ges.configMap.integrations.vmcs.redis_host
}}:{{ .Values.ges.configMap.integrations.vmcs.redis_port | default 10000 }}"

  deployment:
    update_strategy:
      type: RollingUpdate
      rollingUpdate:
        maxUnavailable: 1
        maxSurge: 100%

    envFrom:
      - configMapRef:
          name: "{{.Values.ges.servicename}}-{{.Release.Name}}-configmap"

    min_ready_seconds: 300

    dnsConfig:
      options:
        - name: ndots
          value: "3"

    replicas: 1
```

```
  nodeSelector: {}

  image:
    imagePullSecrets: [  ]
    imagePullPolicy: "Always"
    registry: ""
    name: "genesys/ges"
    version: 9.0.048.00.build.205.rev.55fe5bd94

  hpa:
    enabled: true
    maximumReplicas: 5
    mimimumReplicas: 2
    targetMetrics:
      - type: Resource
        resource:
          name: cpu
          target:
            type: Utilization
            averageUtilization: 50
    behavior:
      scaleDown:
        stabilizationWindowSeconds: 60
        policies:
        - type: Pods
          value: 1
          periodSeconds: 30
      scaleUp:
        stabilizationWindowSeconds: 0
        policies:
        - type: Pods
          value: 1
          periodSeconds: 30
        selectPolicy: Max

priorityClassName: ""

resources:
  limits:
      cpu: "1250m"
      memory: "2048Mi"
  requests:
      cpu: "500m"
      memory: "512Mi"

ingress:
  ingressint:
    enabled: false
  networkPolicies:
    enabled: false

monitoring:
  prometheus:
    enabled: "false"
  newRelic:
    enabled: "false"

podDisruptionBudget:
  enabled: "true"
  strategy:
    minAvailable: 1

tolerations: {}
```

```
secrets:
  configFolders: "/secrets-infra,/secrets-ors,/secrets-gws"
  volumes:
    enabled: true
    list:
    - name: secrets-infra
      secret:
        secretName: ges-secrets-infra
    - name: secrets-ors
      secret:
        secretName: voice-redis-ors-stream
    - name: secrets-gws
      secret:
        secretName: gws-client-credentials
  volumeMounts:
    enabled: true
    list:
      - mountPath: /secrets-infra
        name: secrets-infra
        readOnly: true
      - mountPath: /secrets-ors
        name: secrets-ors
        readOnly: true
      - mountPath: /secrets-gws
        name: secrets-gws
        readOnly: true

service:
  port_internal: 80
  port_external: 8080
  type: "ClusterIP"


servicename: "ges"
```

9. Validate the GES helm charts using:
   ```
   > helm upgrade --install --dry-run -n -f
   ```
   Review the output to ensure that the generated Helm manifest matches your expectations.

Validate the deployment

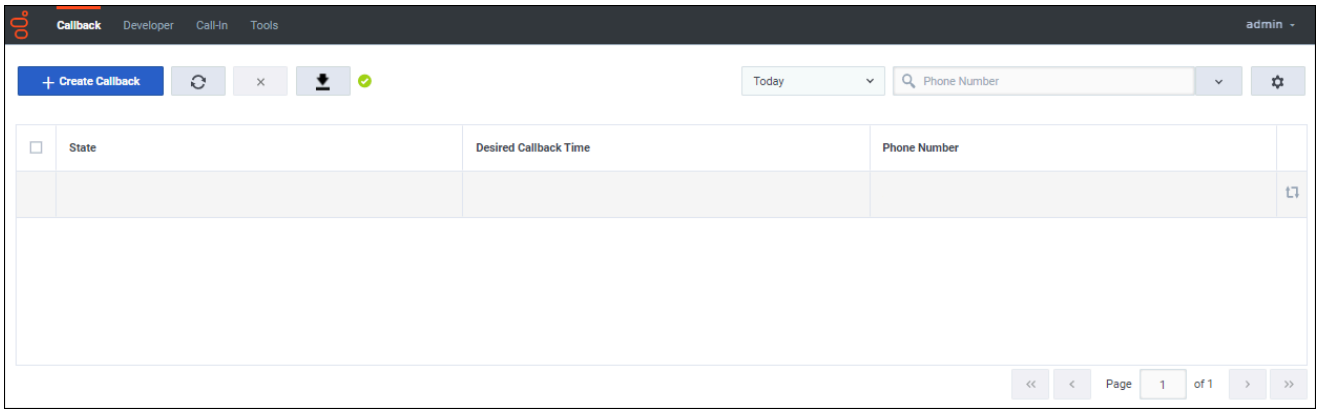
Once GES has been installed and provisioned following the processes outlined above, the best way to validate your deployment is to check that callbacks and other provisioned services work as intended. Logging in to the UI is a good way to check that privileges are working. Also, monitor callbacks, call-ins, and other requests to ensure that they complete successfully.

Validate the UI

Login to the UI.

With a user that has been assigned the **Callback Administrator** role, attempt to log into the Callback UI. You can do this at `/ges`.

If you successfully log in, then you see an empty **Callback** grid as the main feature of the screen:



Navigate to some of the other tabs and verify that no errors pop up in the lower corner of the screen as the system attempts to read and retrieve data.

If, at any point, data fails to load or you are locked out due to insufficient permissions, adjust the roles and privileges assigned to Callback UI users and try again.

Create a callback

There are two ways to create a callback request as an external user. You can create a callback request from the Callback UI or by using the /engagement API. For information about using the Callback UI to create a callback, see Managing callbacks. For information about using the APIs, see Genesys Multicloud CX REST APIs and tutorials for Callback.

Create Call-In request (optional)

The Click-to-Call-In scenario is optional functionality. If you plan to use the Click-to-Call-In feature, see the following pages for more information:

- Description of the Click-to-Call-In feature
- Provision the Click-to-Call-In scenario
- Configure Click-To-Call-In Groups
- View Click-To-Call-In records