Tenant Service Private Edition Guide

Deploy Tenant Service

4/14/2026

Contents

Learn how to deploy Tenant 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 an overview of solution-level deployment, see the deployment tour.

Deployment scenarios

More than one deployment scenario is supported for Tenant Service, including single region, redundant, and multi-region deployment as well as multi-Tenant deployment.

Single region/location/cluster

You deploy Tenant resources in a single Kubernetes cluster within the same or separate namespace (project) with the Voice platform. If shared resources are being deployed across all Tenants, they must also be added to the same target namespace.

The Tenant deployment process creates resources using a *release name* parameter, specified when executing the Helm deployment step. When installed in a single namespace, you must make sure that the release name value is distinct across all Tenants and other deployments.

For example, you might specify the Helm release name in the format `t`. Optionally, if you want the Tenant service name to match other Voice services, you can prefix the Tenant name with `voice-` in the Helm release name. So, in this example, you would specify the release name as `voice-t` during Helm deployment. The value for is the last four characters of the Tenant UUID that you configure in the **values.yaml** file. For more information about the identification parameters for the Tenant service, see Identification.

If you plan to use Prometheus monitoring or Fluent Bit logging framework for Tenant, you must execute the **tenant-monitor** module, as described in tenant-monitor. The module enables the following features:

- Prometheus PodMonitor definition for all tenant pods.
- Common Fluent Bit framework configuration for all tenant pods.

Single Tenant: Basic deployment

Single-node deployment requires a single override file and one "tenant" module to deploy, with reference implementation described at Single service at one location.

To increase the number of nodes, adjust the **node count** parameter. For more information, see Scalability and redundancy parameters.

Upgrade

For information about upgrading Tenant Service, see Upgrade, rollback, or uninstall the Tenant Service.

Multiple Tenants at one location: Basic deployment

You can deploy additional Tenants at the same location using the following guidelines:

- Each Tenant Service must have a unique tenant uuid, shortid, and nickname.
- Each Tenant Service is deployed or upgraded and adjusted independently.

Multiple regions/locations/clusters: Basic deployment

In multi-regional/multi-location deployments, one region/location is considered "master" (from the Tenant perspective) and includes the database backend with `write` capabilities. Other regions/locations have replicas of the database backend in read-only mode. A Tenant Service at each location may be deployed to have one of its nodes running as master (`write` access to provisioning data through the config API) or have all its nodes running only as replicas (`read` access to configuration).

Multi-regional deployments must be performed using the following steps (with prerequisites already satisfied at each region/location):

- If required, deploy the **tenant-monitor** module at a location planned as a Master Tenant node.
- Complete the basic deployment of a Tenant Service in the Master region, including specification of DR parameters for the Master, as per Scalability and redundancy parameters.
- Complete the deployment of the database backend with a replica of the Master database at the location(s) where the replica Tenant nodes are expected to run, including provisioning of access keys/

secrets to access the local replica.

- If required, deploy the **tenant-monitor** module at location(s) where replicas are expected to run.
- Complete the basic Tenant deployment for additional region(s) and specify DR parameters for the Master region (see Scalability and redundancy parameters).

The same customization scenarios described for Tenant nodes can be applied for each location independently.

Upgrade

For information about upgrading Tenant Service, see Upgrade, rollback, or uninstall the Tenant Service.

Deploy the service

This section provides reference commands with key parameters that are required to complete each deployment step.

On this page, the **tenant-values.yaml** file refers to the **values.yaml** file in the Tenant Helm chart. Likewise, the **tenant-monitor-values.yaml** file refers to the **values.yaml** file in the Tenant Monitor Helm chart.

Prerequisites

- Read Before you begin for the full list of prerequisites required to deploy the Tenant Service.
- Mandatory parameter values for basic installation are:
  - tenant uuid (v4)
  - tenant nickname (becomes a Helm release name)
  - all backend parameters (along with all secrets that may be required based on these parameters)
- Before proceeding with the Tenant Service deployment, ensure you have completed procedures in the Configure security section of this guide.
- Ensure you have configured all required overrides in the Helm chart **values.yaml** files, including specifying the correct SIP domain. For information, see Override Helm chart values.

Location-specific deployment steps

tenant-monitor

Monitoring/logging shared configuration and infrastructure deployment:

```
helm upgrade --install --force --wait --timeout 600s -n voice  tenant-monitor https:///tenant-
monitor-$TENANT_MANIFEST_VERSION.tgz  --username "$JFROG_USER" --password "$JFROG_PASSWORD"
```

To enable Prometheus monitoring, you can use the following overrides with tenant-monitor. Use the following changes in the **tenant-monitor-values.yaml** file to implement the changes:

```
prometheus:
  podMonitor:
    create: "true"
```

To enable Fluent Bit to send additional logs to stdout in json format (for selected Tenant functions, such as configuration audit) and/or raw format (such as from internal applications such as StatServer and URS), modify the following changes to upgrade tenant-monitor. Use the following changes in the **tenant-monitor-values.yaml** file to implement the changes.

```
fluent:
  enable: "true"
  rawlogs:
    stdout:
      enable: "true"
  jsonlogs:
    stdout:
      enable: "true"
```

To enable RWX Persistent Volume Claim (PVC) in tenant-monitor to store Tenant logs shared across all Tenant pods, make the following modifications to override values in the **tenant-monitor-values.yaml** file:

```
tenant:
  logging:
    volume:
      enabled: "true"
      createSC: "false"
      createpvClaim: "true"
      logClaim: "tenant-logs-pvc"
      logClaimSize: "5Gi"
      logStorageClass: ""
      Storageprovisioner: "TBD OC provisioner"
      parameters: {}
```

RWX PV is disabled by default; no overrides are required in the **tenant-monitor-values.yaml** file to disable it.

Service-specific deployment steps: Single service at one location

A PostgreSQL database must be available for the Tenant Service before you begin the service deployment. For more information about the database requirements, see Third-party prerequisites. In addition, after the PostgreSQL database is deployed and before you deploy the Tenant Service, you must configure secrets that contain values for certain PostgreSQL database parameters. To configure

the secrets, see Service-specific secrets.

Use the following template if you are deploying with the tenant Helm chart. A single-service deployment can be implemented with the following sample parameters in the **tenant-values.yaml** file:

```
##UUID 4 format ( Set a new UUID for new tenant deployment)
tenantid:


serviceAccount:
    create: true

images:
  imagePullSecrets: mycred
  registry:
  pullPolicy: Always
  tenant:
    tag:

  pgdbInit:
    tag:

  rcsInit:
    tag:
    enable: "true"

  pulseInit:
    tag:
    enable: "true"
    pulseMode: "setup"



tenant:
  general:
    upstreamServices: voice-sipfe:9101,voice-config:9100,ixn-server-{{ $.Values.tenantid
}}:7120,ixn-vqnode-{{ $.Values.tenantid }}:7122"
  pgdb:
    dbhost: "/opt/genesys/dbserver/dbserver"
    dbuser: "/opt/genesys/dbuser/dbuser"
    dbname: "/opt/genesys/dbname/dbname"
  securityContext:
    fsGroup: 0

  logging
 ....
  volumes:
    logPvc:
        enabled: "true"
        logClaimSize: "5Gi"
        accessModes: "ReadWriteOnce"
        logStorageClass: ""   #Replace the storage class with a relevant storage class for
the cluster type

    mounts:
      log:
        - name: log
          mountPath: /opt/genesys/logs/volume
        - name: log
          mountPath: /logs
  secrets:
```

```
pgdb:
  pwd:
    secretName: "/opt/genesys/dbpassword/dbpassword"
    secretKey: "dbpassword"
  volumes: |
    - name: dbpassword
      secret:
        secretName: dbpassword
    - name: dbserver
      secret:
        secretName: dbserver
    - name: dbname
      secret:
        secretName: dbname
    - name: dbuser
      secret:
        secretName: dbuser
  mounts:
    - name: dbpassword
      readOnly: true
      mountPath: "/opt/genesys/dbpassword"
    - name: dbserver
      readOnly: true
      mountPath: "/opt/genesys/dbserver"
    - name: dbname
      readOnly: true
      mountPath: "/opt/genesys/dbname"
    - name: dbuser
      readOnly: true
      mountPath: "/opt/genesys/dbuser"

consul:
  acl:
    secretName: "/opt/genesys/consul-shared-secret/consul-consul-voice-token"
  volumes:
    - name: consul-shared-secret
      secret:
        secretName: consul-voice-token
  mounts:
    - name: consul-shared-secret
      readOnly: true
      mountPath: "/opt/genesys/consul-shared-secret"
redis:
  configPwd:
    secretName: "/opt/genesys/redis-config-secret/redis-config-state"
    volumes:
      - name: redis-config-secret
        secret:
          secretName: redis-config-token
    mounts:
      - name: redis-config-secret
        readOnly: true
        mountPath: "/opt/genesys/redis-config-secret"

  streamPwd:
    secretName: "/opt/genesys/redis-tenant-secret/redis-tenant-stream"
    volumes:
      - name: redis-tenant-secret
        secret:
          secretName: redis-tenant-token
    mounts:
      - name: redis-tenant-secret
        readOnly: true
```

```
          mountPath: "/opt/genesys/redis-tenant-secret"

  kafka:
    pwd:
       secretName: "/opt/genesys/kafka-secrets/kafka-secrets"
    volumes:
      - name: kafka-secrets
        secret:
          secretName: kafka-secrets-token
    mounts:
      - name: kafka-secrets
        mountPath: "/opt/genesys/kafka-secrets"
  gws:
    user:
      secretName: "/opt/genesys/gauth-client-id/clientid"
    pwd:
      secretName: "/opt/genesys/gauth-client-token/clientsecret"
    volumes:
      - name: gauth-client-id
        secret:
          secretName: gauthclientid
      - name: gauth-client-token
        secret:
          secretName: gauthclientsecret
    mounts:
      - name: gauth-client-id
        readOnly: true
        mountPath: "/opt/genesys/gauth-client-id"
      - name: gauth-client-token
        readOnly: true
        mountPath: "/opt/genesys/gauth-client-token"



redis:
  isCluster: true
```

In addition, use the following deployment command:

```
helm upgrade --install --force --wait --timeout 600s -n voice -f ./tenant-node-values.yaml t \
https:///tenant-.tgz \
--username "$JFROG_USER" --password "$JFROG_PASSWORD"
```

The preceding deployment will create a Tenant with the password of the service account set up explicitly and without enabling GWS integration. See Samples and references for values that allow you to reset the Tenant password upon deployment using a pre-generated value from the secret and to enable automated GWS integration.

Samples and references

Enabling a service admin password (the secret should be created as described in the Service account password section):

```
...
tenant:
  serviceuser: "default"
  svcpwdSecretName: "/opt/genesys/service-user-account/svcpassword"
  ...
```

```
  volumes: |
    - name: service-user-account
      secret:
           secretName: svcuseraccount
    ...
  volumeMounts: |
    - name: service-user-account
      readOnly: true
      mountPath: "/opt/genesys/service-user-account"
    ....
  initVolumeMounts: |
    - name: service-user-account
      readOnly: true
      mountPath: "/opt/genesys/service-user-account"
    ....
```

To enable stdout log output for all Tenant components, make the following modifications to override values in the **tenant-values.yaml** file.

```
images
fbregistry: fluent/fluent-bit
...
  fluentBit:
    enable: "true"
    name: json-sidecar
    tag: 1.8.x

  fluentBitUrs:
    enable: "true"
    name: stdouturs-sidecar
    tag: 1.8.x

  fluentBitSs:
    enable: "true"
    name: stdoutss-sidecar
    tag: 1.8.x

  fluentBitOcs:
    enable: "true"
    name: stdoutocs-sidecar
    tag: 1.8.x

  fluentBitCs:
    enable: "true"
    name: stdoutcs-sidecar
    tag: 1.8.x

tenant:
 ...
  logging:
    volumes:
      log:
        - name: log
      jsonLog:
        - name: fluent-logs
          emptyDir: {}
      stdoutUrsLog:
        - name: fluenturs-logs
          emptyDir: {}
      stdoutOcsLog:
        - name: fluentocs-logs
          emptyDir: {}
      stdoutSsLog:
```

```
      - name: fluentss-logs
        emptyDir: {}
    stdoutCsLog:
      - name: fluentcs-logs
        emptyDir: {}
    fluentBconfigmap:
      - configMap:
          defaultMode: 420
          name: tenants-fluent-bit-config
        name: tenants-fluent-bit-config
    fluentBconfigmapCs:
      - configMap:
          defaultMode: 420
          name: tenants-fluent-bit-config-cs
        name: tenants-fluent-bit-config-cs
    fluentBconfigmapSs:
      - configMap:
          defaultMode: 420
          name: tenants-fluent-bit-config-ss
        name: tenants-fluent-bit-config-ss
    fluentBconfigmapOcs:
      - configMap:
          defaultMode: 420
          name: tenants-fluent-bit-config-ocs
        name: tenants-fluent-bit-config-ocs
    fluentBconfigmapUrs:
      - configMap:
          defaultMode: 420
          name: tenants-fluent-bit-config-urs
        name: tenants-fluent-bit-config-urs
  mounts:
    log:
      - name: log
        mountPath: /opt/genesys/logs/volume
      - name: log
        mountPath: /logs
    jsonLog:
      - name: fluent-logs
        mountPath: "/opt/genesys/logs/JSON"
    stdoutUrsLog:
      - name: fluenturs-logs
        mountPath: "/opt/genesys/logs/URS"
    stdoutSsLog:
      - name: fluentss-logs
        mountPath: "/opt/genesys/logs/SS"
    stdoutOcsLog:
      - name: fluentocs-logs
        mountPath: "/opt/genesys/logs/OCS"
    stdoutCsLog:
      - name: fluentcs-logs
        mountPath: "/opt/genesys/logs/confserv"

    fbJsonLog:
      - name: fluent-logs
        mountPath: "/mnt/logs"
    fbstdoutUrsLog:
      - name: fluenturs-logs
        mountPath: "/mnt/logs"
    fbstdoutSsLog:
      - name: fluentss-logs
        mountPath: "/mnt/logs"
    fbstdoutOcsLog:
      - name: fluentocs-logs
```

```
        mountPath: "/mnt/logs"
    fbstdoutCsLog:
      - name: fluentcs-logs
        mountPath: "/mnt/logs"

    fluentBconfigmap:
        - mountPath: /fluent-bit/etc/
          name: tenants-fluent-bit-config
    fluentBconfigmapCs:
        - mountPath: /fluent-bit/etc/
          name: tenants-fluent-bit-config-cs
    fluentBconfigmapSs:
        - mountPath: /fluent-bit/etc/
          name: tenants-fluent-bit-config-ss
    fluentBconfigmapOcs:
        - mountPath: /fluent-bit/etc/
          name: tenants-fluent-bit-config-ocs
    fluentBconfigmapUrs:
        - mountPath: /fluent-bit/etc/
          name: tenants-fluent-bit-config-urs
```

You can deploy Persistent Volume/Persistent Volume Claim (PV/PVC) in two ways:

1. Enable ReadWriteOnce (RWO) from the tenant Helm chart, which maintains unique PVCs for each pod/ replica from the same Tenant.
2. Enable ReadWriteMany (RWX) from the tenant-monitor Helm chart, which has multiple Tenant pods sharing the same PVC.

To enable RWO PV/PVC logging from individual Tenant pods in Statefulset, make the following modifications to override the values in the **tenant-values.yaml** file. RWO Persistent Volume is disabled by default.

```
logging
 ....
  volumes:
    logPvc:
        enabled: "true"
        logClaimSize: "5Gi"
        accessModes: "ReadWriteOnce"
        logStorageClass: ""   #Replace the storage class that's relevant to the Openshift
Cluster
```

Enabling GWS integration (the secret should be created as described in the Genesys Authentication backend secrets section):

```
tenant:
…
gws:
    # enable: Enable GWS registeration about tenant
    # tls: Enable/Disable Secure connection to GWS
    # authEndpoint: GWS auth end point
    # envEndpoint: GWS env end point for Registeration
    # db: Pass DB infromation for GWS to connect to PSQL DB for read and store data
    enable: "true"
    tls: false
    authEndpoint: "gauth-auth.gauth.svc.cluster.local"
    envEndpoint: ""
    db:
      enable: "false"
      read: "false"
```

```
      init: "false"
             secrets:
……….
gws:
      enabled: true
      user:
        secretName: "/opt/genesys/gauth-client-id/clientid"
      pwd:
        secretName: "/opt/genesys/gauth-client-token/clientsecret"
      volumes:
        - name: gauth-client-id
          secret:
            secretName: gauthclientid
        - name: gauth-client-token
          secret:
            secretName: gauthclientsecret
      mounts:
        - name: gauth-client-id
          readOnly: true
          mountPath: "/opt/genesys/gauth-client-id"
        - name: gauth-client-token
          readOnly: true
          mountPath: "/opt/genesys/gauth-client-token"
```

To mount the PVC to store Tenant logs, make the following modifications to override the values in the **tenant-values.yaml** file:

```
.....

tenant:
  ...
    logging:
    mounts:
      log:
        - name: log
          mountPath: /opt/genesys/logs/volume
        - name: log
          mountPath: /logs
```

Validate the deployment

Content coming soon