Voice Microservices Private Edition Guide

Deploy Voice Microservices

4/9/2026

Contents

- 1 Assumptions
- 2 General deployment prerequisites
- 3 Deployment order for Voice Microservices
- 4 Create the Voice namespace
- 5 Deploy Voice services
  - 5.1 Storage class and Claim name
  - 5.2 Configure the DNS Server for voice-sip
- 6 Voice Service Helm chart deployment
- 7 Deploy the Tenant service
- 8 Validate the deployment

Learn how to deploy Voice Microservices 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 Voice Microservices.
>
> To deploy the Tenant service, see the *Tenant Service Private Edition Guide*.
>
> For information about deploying Voicemail Service, see Deploy Voicemail.

General deployment prerequisites

Before you deploy the Voice Services, you must deploy the infrastructure services. See Third-party prerequisites for the list of required infrastructure services.

In addition, see Consul requirements for Voice services and Redis requirements for Voice services for information about specific configuration that must be completed in Consul before you configure or deploy Voice Microservices.

To override values for both the infrastructure services and voice services, see Override Helm chart values.

Deployment order for Voice Microservices

Genesys recommends the following order of deployment for the Voice Microservices:

- Voice Services
- Tenant Service
- Voicemail Service

Create the Voice namespace

Before deploying Voice Services and their dependencies, create a namespace using the following command:

```
kubectl create ns voice
```

In all Voice Services and the configuration files of their dependencies, the namespace is **voice**. If you want a specific, custom namespace, create the namespace (using the preceding command) and remember to change the namespace in files, as required.

Deploy Voice services

Voice Services require a Persistent Volume Claim (PVC); the Voice SIP Cluster Service uses a persistent volume to store traditional SIP Server logs. Before deploying Voice Services, create the PVC.

Storage class and Claim name

The created persistent volume must be configured in the **sip_node_override_values.yaml** file as shown below:

```
# pvc will be created for logs
volumes:
  pvcLog:
    create: true
    claim: sip-log-pvc
    storageClass: voice
    volumeName:  (ex sip-log-pv)

  pvcJsonLog:
    create: true
    claim: sip-json-log-pvc
    storageClass: voice
    volumeName:  (ex sip-log-pv)
```

Configure the DNS Server for voice-sip

The Voice SIP Cluster Service requires the DNS server to be configured in its **sip_node_override_values.yaml** file. Follow the steps in the Kubernetes documentation to install a **dnsutils** pod. Using the **dnsutils** pod, get the **dnsserver** that's used in the environment.

The default value in the SIP Helm chart is `10.0.0.10`. If the **dnsserver** address is different, update it in the **sip_node_override_values.yaml** file as shown below:

```
# update dns server ipaddress
  context:
    envs:
      dnsServer: "10.202.0.10"
```

Voice Service Helm chart deployment

Deploy the Voice Services using the provided Helm charts.

```
helm upgrade --install --force --wait --timeout 300s -n voice -f ./voice_helm_values/
agent_override_values.yaml voice-agent /voice-agent-.tgz --set version= --username
"$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 300s -n voice -f ./voice_helm_values/
callthread_override_values.yaml voice-callthread /voice-callthread-.tgz --set version= --
username "$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 200s -n voice -f ./voice_helm_values/
config_override_values.yaml voice-config /voice-config-.tgz --set version= --username
"$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 300s -n voice -f ./voice_helm_values/
dialplan_override_values.yaml voice-dialplan /voice-dialplan-.tgz --set version= --username
"$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 200s -n voice -f ./voice_helm_values/
ors_node_override_values.yaml voice-ors /voice-ors-.tgz --set version= --username
"$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 300s -n voice -f ./voice_helm_values/
registrar_override_values.yaml voice-registrar /voice-registrar-.tgz --set version= --
username "$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 200s -n voice -f ./voice_helm_values/
rq_node_override_values.yaml voice-rq /voice-rq-.tgz --set version= --username "$JFROG_USER"
--password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 200s -n voice -f ./voice_helm_values/
sip_node_override_values.yaml voice-sip /voice-sip-.tgz --set version= --username
"$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 300s -n voice -f ./voice_helm_values/
sipfe_override_values.yaml voice-sipfe /voice-sipfe-.tgz --set version= --username
"$JFROG_USER" --password "$JFROG_PASSWORD"

helm upgrade --install --force --wait --timeout 300s -n voice -f ./voice_helm_values/
sipproxy_override_values.yaml voice-sipproxy /voice-sipproxy-.tgz --set version= --username
"$JFROG_USER" --password "$JFROG_PASSWORD"
```

The following table contains a list of the minimum recommended Helm chart versions that should be used:

Service name	Helm chart version
voice-config	voice-config-9.0.11.tgz
voice-dialplan	voice-dialplan-9.0.08.tgz
voice-registrar	voice-registrar-9.0.14.tgz
voice-agent	voice-agent-9.0.10.tgz
voice-callthread	voice-callthread-9.0.12.tgz
voice-sip	voice-sip-9.0.22.tgz
voice-sipfe	voice-sipfe-9.0.06.tgz
voice-sipproxy	voice-sipproxy-9.0.09.tgz
voice-rq	voice-rq-9.0.08.tgz
voice-ors	voice-ors-9.0.08.tgz

Deploy the Tenant service

The Tenant Service is included with the Voice Microservices, but has its own deployment procedure. To deploy the Tenant Service, see Deploy the Tenant Service.

Validate the deployment

Follow the steps below to validate the successful deployment of voice microservices.

1. Verify the helm deployments using the following command.

   ```
   helm list -n voice
   ```

   Sample output:

   ```
   NAME                                        NAMESPACE       REVISION
    UPDATED                                  STATUS          CHART

    APP VERSION
    voice-agent-latest                                  voice           4
    2022-08-18 13:22:12.355810905 +0000 UTC deployed        voice-
    agent-100.0.1000006           1.0
    voice-callthread-latest                             voice           70
    2022-08-18 09:44:07.078583581 +0000 UTC deployed        voice-
    callthread-100.0.1000006      1.0
    voice-config-latest                                 voice           61
    2022-08-19 01:33:02.039668264 +0000 UTC deployed        voice-
    config-100.0.1000006          1.0
    voice-dialplan-latest                               voice           5
    2022-08-18 12:33:31.223393121 +0000 UTC deployed        voice-
    dialplan-100.0.1000009        1.0
    voice-ors-latest                                    voice           1
   ```

```
2022-08-15 21:40:32.013855856 +0000 UTC deployed        voice-
ors-100.0.1000018             1.0
voice-registrar-latest                          voice           108
2022-08-18 13:41:26.37007884 +0000 UTC  deployed        voice-
registrar-100.0.1000007       latest-aa9f28a
voice-rq-latest                                 voice           14
2022-08-18 13:44:07.187279228 +0000 UTC deployed        voice-
rq-100.0.1000004              1.0
voice-sip-latest                                voice           193
2022-08-10 23:06:05.057511521 +0000 UTC deployed        voice-
sip-100.0.1000018             1.0
voice-sipfe-latest                              voice           73
2022-08-10 23:49:45.166013304 +0000 UTC deployed        voice-
sipfe-100.0.1000006           1.0
voice-sipproxy-latest                           voice           5
2022-08-11 17:13:30.894221491 +0000 UTC deployed        voice-
sipproxy-100.0.1000007        1.0
voice-voicemail-latest                          voice           67
2022-08-18 15:18:47.347509225 +0000 UTC deployed        voice-
voicemail-100.0.1000015       1.0
```

2. Verify readiness state of Kubernetes objects using the kubectl commands.
    1. Run the following command to check the deployments:

        ```
        kubectl get deployments -n voice
        ```

        Sample output:

        ```
        NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
        voice-agent           2/2     2            2           40d
        voice-callthread      3/3     3            3           704d
        voice-config          1/1     1            1           704d
        voice-dialplan        1/1     1            1           41d
        voice-registrar       1/1     1            1           703d
        voice-sip-debug-kpan  2/2     2            2           68d
        voice-sipfe           3/3     3            3           727d
        voice-voicemail       1/1     1            1           87d
        ```

    2. Run the following command to check the Statefulsets:

        ```
        kubectl get statefulset -n voice
        ```

        Sample output:

        ```
        NAME             READY   AGE
        voice-ors        50/50   40d
        voice-rq         20/20   40d
        voice-sip        30/30   703d
        voice-sipproxy   5/5     40d
        ```

3. Check if all the pods are running and in Ready state.
    1. Run the following command to check the readiness of the pods.

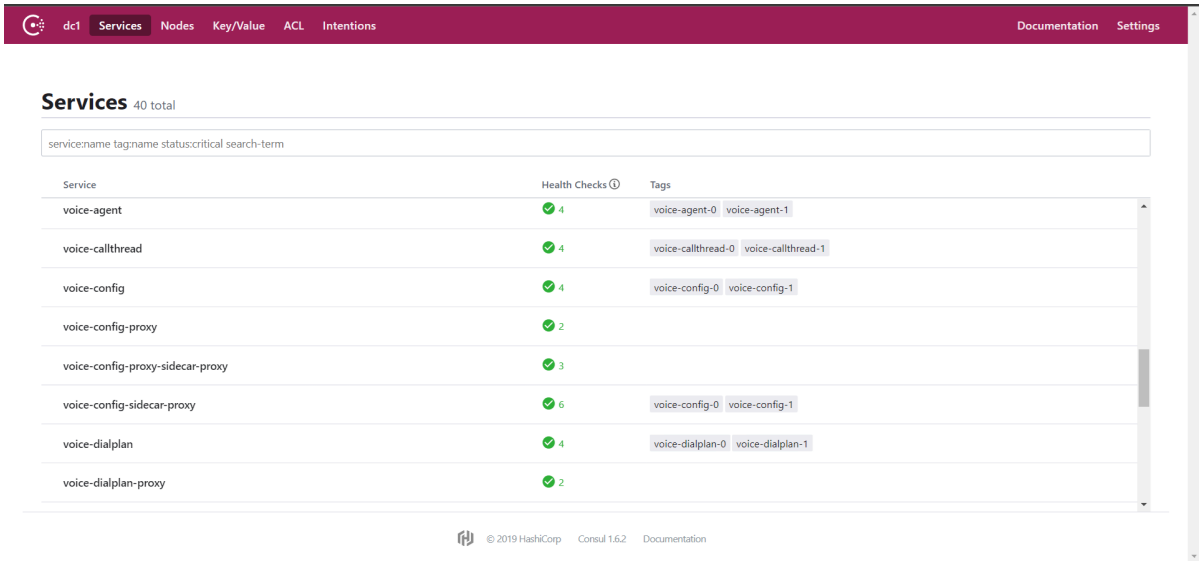
        ```
        kubectl get pods -n voice
        ```

        Sample output:

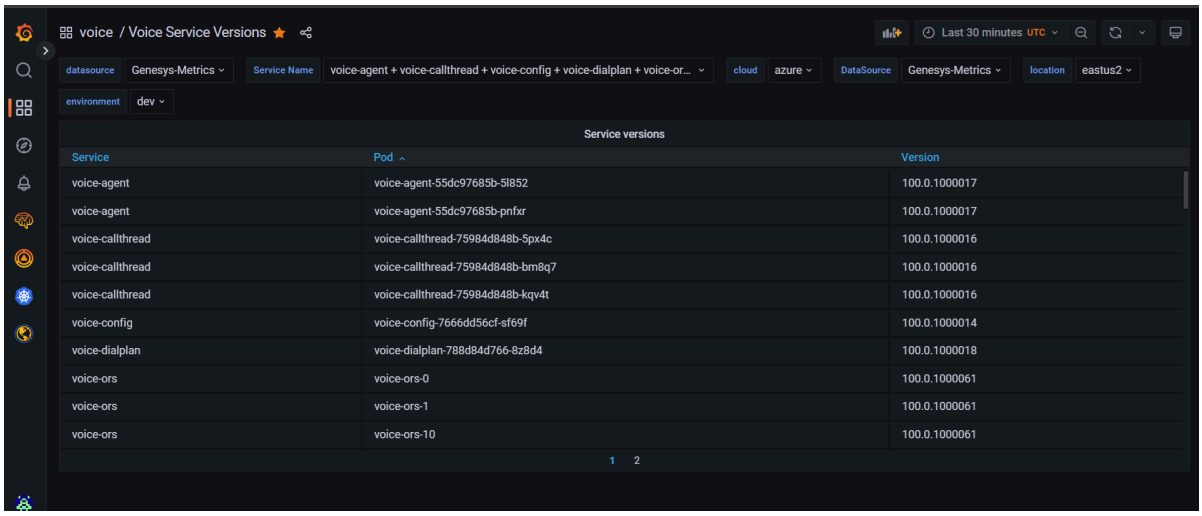
        ```
        NAME                                READY   STATUS        RESTARTS        AGE
        t2100-0                             3/3     Running       0
        4d23h
        voice-agent-55dc97685b-pnfxr        2/2     Running       0
        ```

```
170m
voice-callthread-75984d848b-bm8q7       2/2     Running       0
170m
voice-callthread-75984d848b-kqv4t       2/2     Running       0
170m
voice-config-7666dd56cf-sf69f           2/2     Running       0              39h
voice-dialplan-788d84d766-8z8d4         2/2     Running       0              37h
voice-ors-0                             2/2     Running       0              18h
voice-ors-1                             2/2     Running       0
6d5h
voice-registrar-6c54c6bc9-tkvk2         2/2     Running       0              39h
voice-rq-0                              2/2     Running       0              38h
voice-rq-1                              2/2     Running       0
4d17h
voice-sip-0                             3/3     Running       0              39h
voice-sip-1                             3/3     Running       0              11d
voice-sipfe-56c7bc77dd-7fpkh            2/2     Running       0
170m
voice-sipproxy-0                        2/2     Running       0              11d
voice-voicemail-66f745448b-wqmfc        2/2     Running       0
4d20h
```

4. Verify the health status of the pods in Consul dashboard.
   If the services are running and in Ready state, the health check will be marked as Green in Consul dashboard.



5. Check the versions of microservices in Grafana dashboard.
   Only if voice-dashboards are deployed in the voice namespace, you can perform this check in the dashboard.



6. Check for any crash, KafkaJS or Redis connection errors in Prometheus, Grafana dashboards and/or logs of the respective microservices.

From a functional point of view, you can validate the voice microservices deployment by performing the following steps.

1. Before you can validate the voice microservices, you must create few objects in the Tenant configdb to start the verification.
    1. Port forward the Tenant instance at 8888 port and access the tenant objects through Configuration Manager application.

        ```
        kubectl port forward t2100-0 8888:8888 -n voice
        ```

    2. Create a few Directory Numbers (DNs) under the Sip_Cluster switch with the following options:

        ```
        [TServer]
                contact=*
                dual-dialog-enabled=false
                infra-class=2
                make-call-rfc3725-flow=1
                refer-enabled=false
                sip-cti-control=talk,hold
                sip-ring-tone-mode=1
                use-contact-as-dn=true
                use-register-for-service-state=false
        ```

    3. Create a Place object and map the DNs created.
    4. Create new Agents with username and password, under the "Persons" section.
    5. Map the Place to the agent.
2. Once the objects are created successfully, follow the steps below to validate the voice microservices deployment..
    1. Register the DNs from Endpoints.
    2. Login/Logout the Agents from Workspace Web Edition or a similar application and change the states - Ready, Not Ready and Logout.
    3. Make few test calls between the agents.

4. Perform other call functionalities like - hold/retrieve, conference, transfer, after call work, and so on.
5. If Designer is available, load different strategies onto route points (external facing SBC Numbers) and validate if the inbound call made from PSTN is being routed to the agent/skill group configured.

3. Additionally, you can also check the below after the deployment of voice microservices.
    1. Verify whether the Grafana dashboards of the voice microservices are updated with relevant data and they reflect the status of the services correctly.
    2. Check if the alerts and alarms are configured for the voice microservices and are active.