Genesys Voice Platform Private Edition Guide

Deploy Genesys Voice Platform

8/19/2022

Contents

Learn how to deploy Genesys Voice Platform (GVP) 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 or OpenShift project, 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.

Deploy

> **Important**
> Make sure to review Before you begin for the full list of prerequisites required to deploy Genesys Voice Platform.

Prerequisites

- Consul with Service Mesh and DNS
- Availability of shared Postgres for GVP Configuration Server

- Availability of SQL Server database for Reporting Server
  - Create DB in advance (for example, DB Name: **gvp_rs**).
  - There is a requirement for one user to have admin or database owner (dbo) access and a second user with read only (ro) access.
  - Use these credentials for the creation of Reporting Server secrets.

Environment setup

OpenShift

- Log in to the OpenShift cluster from the remote host via CLI.

```
oc login --token  --server
```

- Check the cluster version.

```
oc get clusterversion
```

- Create gvp project in the OpenShift cluster.

```
oc new-project gvp
```

- Set the default project to GVP.

```
oc project gvp
```

- Bind security context constraints (SCC) to the Genesys user using default service account.

```
oc adm policy add-scc-to-user genesys-restricted -z default -n gvp
```

- Create a secret for docker-registry to pull an image from JFrog.

```
oc create secret docker-registry  --docker-server= --docker-username= --docker-password= --
docker-email=
```

- Link the secret to the default service account with pull role.

```
oc secrets link default  --for=pull
```

GKE

- Log in to the gke cluster.

```
gcloud container clusters get-credentials gke1
```

- Create gvp project in gke cluster using following manifest file.

create-gvp-namespace.json

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

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

- Confirm namespace creation.

```
kubectl describe namespace gvp
```

The order of installation matters with GVP. To deploy without errors, install in this order:

1. GVP Configuration Server

2. GVP ServiceDiscovery

3. GVP Reporting Server

4. GVP Resource Manager

5. GVP Media Control Platform

Helm chart release URLs

Download the GVP Helm charts from JFrog using your credentials:

gvp-configserver : https:///gvp-configserver-.tgz

gvp-sd : https:///gvp-sd-.tgz

gvp-rs : https:///gvp-rs-.tgz

gvp-rm : https:///gvp-rm-.tgz

gvp-mcp : https:///gvp-mcp-.tgz

For version numbers, refer to Helm charts and containers for Genesys Voice Platform.

1. GVP Configuration Server

Secrets creation

Create the following secrets that are required for the service deployment.

postgres-secret

db-hostname: Hostname of DB server

db-name: Database name

db-password: Password for DB user

db-username: Username for DB

server-name: Hostname of DB server

```
apiVersion: v1
kind: Secret
metadata:
  name: postgres-secret
  namespace: gvp
type: Opaque
data:
  db-username:
  db-password:
  db-hostname: cG9zdGdyZXMtcncuaW5mcmEuc3ZjLmNsdXN0ZXIubG9jYWw=
  db-name: Z3Zw
  server-name: cG9zdGdyZXMtcncuaW5mcmEuc3ZjLmNsdXN0ZXIubG9jYWw=
```

Run the following command:

```
kubectl apply -f postgres-secret.yaml
```

configserver-secret

password: Password to set for Config DB

username: Username to set for Config DB

```
apiVersion: v1
kind: Secret
metadata:
```

```
  name: configserver-secret
  namespace: gvp
type: Opaque
data:
  username:
  password:
```

Run the following command:

```
kubectl apply -f configserver-secret.yaml
```

Install Helm chart

Download the required Helm chart release from the JFrog repository and install. Refer to Helm Chart URLs.

```
helm install gvp-configserver ./ -f gvp-configserver-values.yaml
```

Set the following values in your values.yaml for Configuration Server:

priorityClassName >> Set to a priority class that exists on the cluster (or create it instead).

imagePullSecrets >> Set to your pull secret name.

gvp-configserver-values.yaml

```
# Default values for gvp-configserver.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

## Global Parameters
## Add labels to all the deployed resources
##
podLabels: {}

## Add annotations to all the deployed resources
##
podAnnotations: {}

serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name:

## Deployment Configuration
## replicaCount should be 1 for Config Server
replicaCount: 1

## Base Labels. Please do not change these.
serviceName: gvp-configserver
component: shared
# Namespace
```

```
partOf: gvp

## Container image repo settings.
image:
  confserv:
    registry: pureengage-docker-staging.jfrog.io
    repository: gvp/gvp_confserv
    pullPolicy: IfNotPresent
    tag: "{{ .Chart.AppVersion }}"
  serviceHandler:
    registry: pureengage-docker-staging.jfrog.io
    repository: gvp/gvp_configserver_servicehandler
    pullPolicy: IfNotPresent
    tag: "{{ .Chart.AppVersion }}"
  dbInit:
    registry: pureengage-docker-staging.jfrog.io
    repository: gvp/gvp_configserver_configserverinit
    pullPolicy: IfNotPresent
    tag: "{{ .Chart.AppVersion }}"

## Config Server App Configuration
configserver:
  ## Settings for liveness and readiness probes
  ## !!! THESE VALUES SHOULD NOT BE CHANGED UNLESS INSTRUCTED BY GENESYS !!!
  livenessValues:
    path: /cs/liveness
    initialDelaySeconds: 30
    periodSeconds: 60
    timeoutSeconds: 20
    failureThreshold: 3
    healthCheckAPIPort: 8300

  readinessValues:
    path: /cs/readiness
    initialDelaySeconds: 30
    periodSeconds: 30
    timeoutSeconds: 20
    failureThreshold: 3
    healthCheckAPIPort: 8300

  alerts:
    cpuUtilizationAlertLimit: 70
    memUtilizationAlertLimit: 90
    workingMemAlertLimit: 7
    maxRestarts: 2

## PVCs defined
#  none

## Define service(s) for application
service:
  type: ClusterIP
  host: gvp-configserver-0
  port: 8888
  targetPort: 8888

## Service Handler configuration.
serviceHandler:
  port: 8300

## Secrets storage related settings - k8s secrets only
secrets:
  # Used for pulling images/containers from the respositories.
```

```
  imagePull:
    - name: pureengage-docker-dev
    - name: pureengage-docker-staging

  # Config Server secrets. If k8s is false, csi will be used, else k8s will be used.
  # Currently, only k8s is supported!
  configServer:
    secretName: configserver-secret
    secretUserKey: username
    secretPwdKey: password
    #csiSecretProviderClass: keyvault-gvp-gvp-configserver-secret

  # Config Server Postgres DB secrets and settings.
  postgres:
    dbName: gvp
    dbPort: 5432
    secretName: postgres-secret
    secretAdminUserKey: db-username
    secretAdminPwdKey: db-password
    secretHostnameKey: db-hostname
    secretDbNameKey: db-name
    #secretServerNameKey: server-name

## Ingress configuration
ingress:
  enabled: false
  annotations: {}
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
  hosts:
    - host: chart-example.local
      paths: []
  tls: []
  #  - secretName: chart-example-tls
  #    hosts:
  #      - chart-example.local

## App resource requests and limits
## ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  requests:
    memory: "512Mi"
    cpu: "500m"
  limits:
    memory: "1Gi"
    cpu: "1"

## App containers' Security Context
## ref: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/#set-the-
security-context-for-a-container
##
## Containers should run as genesys user and cannot use elevated permissions
##
securityContext:
  runAsUser: 500
  runAsGroup: 500
  # capabilities:
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000
```

```
podSecurityContext: {}
  # fsGroup: 2000

## Priority Class
## ref: https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption/
## NOTE: this is an optional parameter
##
priorityClassName: system-cluster-critical

## Affinity for assignment.
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-
affinity
##
affinity: {}

## Node labels for assignment.
## ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Tolerations for assignment.
## ref: https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/
##
tolerations: []

## Service/Pod Monitoring Settings
## Whether to create Prometheus alert rules or not.
prometheusRule:
  create: true

## Grafana dashboard Settings
## Whether to create Grafana dashboard or not.
grafana:
  enabled: true

## Enable network policies or not
networkPolicies:
  enabled: false

## DNS configuration options
dnsConfig:
  options:
    - name: ndots
      value: "3"
```

Verify the deployed resources

Verify the deployed resources from the OpenShift console/CLI.

2. GVP Service Discovery

NOTE: After GVP-SD (Service Discovery) pod gets deployed, you will notice a few errors. Please ignore them and move on to the next deployment. This will start working once Resource Manager (RM) and Media Control Platform (MCP) are deployed.

Secrets creation

Create the following secrets that are required for the service deployment.

shared-consul-consul-gvp-token

shared-consul-consul-gvp-token-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: shared-consul-consul-gvp-token
  namespace: gvp
type: Opaque
data:
  consul-consul-gvp-token: ZmU2NjFkNWYtYzVmNi1mZTJlLTgyM2MtYTAyZGQwN2JlMzll
```

Run the following command:

```
kubectl create -f shared-consul-consul-gvp-token-secret.yaml
```

ConfigMap creation

Create the following ConfigMap that is required for the service deployment.

Caveat

If the tenant has not been deployed yet, then you will not have the information needed to populate the config map. An empty config-map can be created using:

```
kubectl create configmap tenant-inventory -n gvp
```

Create Config based on Tenant provisioning via Service Discovery Container.

t100.json

```
{
    "name": "t100",
    "id": "80dd",
    "gws-ccid": "9350e2fc-a1dd-4c65-8d40-1f75a2e080dd",
    "default-application": "IVRAppDefault"
}
```

Run the following command:

Add Config Map

```
kubectl create configmap tenant-inventory --from-file t100.json -n gvp
```

Install Helm chart

Download the required Helm chart release from the JFrog repository and install. Refer to Helm Chart

URLs.

```
helm install gvp-sd ./ -f gvp-sd-values.yaml
```

gvp-sd-values.yaml

```
# Default values for gvp-sd.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

## Global Parameters
## Add labels to all the deployed resources
##
podLabels: {}

## Add annotations to all the deployed resources
##
podAnnotations: {}

serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name:

## Deployment Configuration
replicaCount: 1
smtp: allowed

## Name overrides
nameOverride: ""
fullnameOverride: ""

## Base Labels. Please do not change these.
component: shared
partOf: gvp

image:
  registry: pureengage-docker-staging.jfrog.io
  repository: gvp/gvp_sd
  tag: "{{ .Chart.AppVersion }}"
  pullPolicy: IfNotPresent

## PVCs defined
# none

## Define service for application.
service:
  name: gvp-sd
  type: ClusterIP
  port: 8080

## Application configuration parameters.
env:
  MCP_SVC_NAME: "gvp-mcp"
  EXTERNAL_CONSUL_SERVER: ""
```

```
  CONSUL_PORT: "8501"
  CONFIG_SERVER_HOST: "gvp-configserver"
  CONFIG_SERVER_PORT: "8888"
  CONFIG_SERVER_APP: "default"
  HTTP_SERVER_PORT: "8080"
  METRICS_EXPORTER_PORT: "9090"
  DEF_MCP_FOLDER: "MCP_Configuration_Unit\\MCP_LRG"
  TEST_MCP_FOLDER: "MCP_Configuration_Unit_Test\\MCP_LRG"
  SYNC_INIT_DELAY: "10000"
  SYNC_PERIOD: "60000"
  MCP_PURGE_PERIOD_MINS: "0"
  EMAIL_METERING_FACTOR: "10"
  RECORDINGS_CONTAINER: "ccerp-recordings"
  TENANT_KV_FOLDER: "tenants"
  TENANT_CONFIGMAP_FOLDER: "/etc/config"
  SMTP_SERVER: "smtp-relay.smtp.svc.cluster.local"

## Secrets storage related settings
secrets:
  # Used for pulling images/containers from the respositories.
  imagePull:
    - name: pureengage-docker-dev
    - name: pureengage-docker-staging

  # If k8s is true, k8s will be used, else vault secret will be used.
  configServer:
    k8s: true
    k8sSecretName: configserver-secret
    k8sUserKey: username
    k8sPasswordKey: password
    vaultSecretName: "/configserver-secret"
    vaultUserKey: "configserver-username"
    vaultPasswordKey: "configserver-password"

  # If k8s is true, k8s will be used, else vault secret will be used.
  consul:
    k8s: true
    k8sTokenName: "shared-consul-consul-gvp-token"
    k8sTokenKey: "consul-consul-gvp-token"
    vaultSecretName: "/consul-secret"
    vaultSecretKey: "consul-consul-gvp-token"

  # GTTS key, password via k8s secret, if k8s is true.  If false, this data comes from tenant
profile.
  gtts:
    k8s: false
    k8sSecretName: gtts-secret
    EncryptedKey: encrypted-key
    PasswordKey: password

ingress:
  enabled: false
  annotations: {}
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
  hosts:
    - host: chart-example.local
      paths: []
  tls: []
  #  - secretName: chart-example-tls
  #    hosts:
  #      - chart-example.local
```

```
resources:
  requests:
    memory: "2Gi"
    cpu: "1000m"
  limits:
    memory: "2Gi"
    cpu: "1000m"

## App containers' Security Context
## ref: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/#set-the-
security-context-for-a-container
##
## Containers should run as genesys user and cannot use elevated permissions
## Pod level security context
podSecurityContext:
  fsGroup: 500
  runAsUser: 500
  runAsGroup: 500
  runAsNonRoot: true

## Container security context
securityContext:
  runAsUser: 500
  runAsGroup: 500
  runAsNonRoot: true

## Priority Class
## ref: https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption/
## NOTE: this is an optional parameter
##
priorityClassName: system-cluster-critical

## Affinity for assignment.
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-
affinity
##
affinity: {}

## Node labels for assignment.
## ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Tolerations for assignment.
## ref: https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/
##
tolerations: []

## Service/Pod Monitoring Settings
prometheus:
  # Enable for Prometheus operator
  podMonitor:
    enabled: true

## Enable network policies or not
networkPolicies:
  enabled: false

## DNS configuration options
dnsConfig:
  options:
    - name: ndots
      value: "3"
```

Verify the deployed resources

Verify the deployed resources from the OpenShift console/CLI.

3. GVP Reporting Server

Secrets creation

Create the following secrets that are required for the service deployment.

rs-dbreader-password

db_hostname: Hostname of DB server

db_name: Database name

db_password: Password for DB user

db_username: Username for DB

rs-dbreader-password-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: rs-dbreader-password
  namespace: gvp
type: Opaque
data:
  db_username:
  db_password:
  db_hostname: bXNzcWxzZXJ2ZXJvcGVuc2hpZnQuZGF0YWJhc2Uud2luZG93cy5uZXQ=
  db_name: cnNfZ3Zw
```

Run the following command:

```
kubectl create -f rs-dbreader-password-secret.yaml
```

shared-gvp-rs-sqlserer-secret

db-admin-password: Password for DB admin

db-reader-password: Password for reader

shared-gvp-rs-sqlserer-secret.yaml

```
apiVersion: v1
```

```
kind: Secret
metadata:
  name: shared-gvp-rs-sqlserver-secret
  namespace: gvp
type: Opaque
data:
  db-admin-password:
  db-reader-password:
```

Run the following command:

```
kubectl create -f shared-gvp-rs-sqlserer-secret.yaml
```

Persistent Volumes creation

Create the following Persistent Volumes (PVs) that are required for the service deployment.

gvp-rs-0

gvp-rs-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: gvp-rs-0
namespace: gvp
spec:
capacity:
storage: 30Gi
accessModes:
- ReadWriteOnce
persistentVolumeReclaimPolicy: Retain
storageClassName: gvp
nfs:
path:  /export/vol1/PAT/gvp/rs-01
server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-rs-pv.yaml
```

Install Helm chart

Download the required Helm chart release from the JFrog repository and install. Refer to Helm Chart URLs.

```
helm install gvp-rs ./ -f gvp-rs-values.yaml
```

Set the following values in your values.yaml:

- priorityClassName >> Set to a priority class that exists on the cluster (or create it instead).
- imagePullSecrets >> Set to your pull secret name.
- keyVaultSecret: false >> Make sure this is false to force use of k8s secrets.
- storageClass: genesys-gvp >> Set to your storage class.

gvp-rs-values.yaml

```
## Global Parameters
## Add labels to all the deployed resources
##
labels:
  enabled: true
  serviceGroup: "gvp"
  componentType: "shared"

serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name:

## Primary App Configuration
##
# primaryApp:
# type: ReplicaSet
# Should include the defaults for replicas
deployment:
  replicaCount: 1
  strategy: Recreate
  namespace: gvp
nameOverride: ""
fullnameOverride: ""

image:
  registry: pureengage-docker-staging.jfrog.io
  gvprsrepository: gvp/gvp_rs
  snmprepository: gvp/gvp_snmp
  rsinitrepository: gvp/gvp_rs_init
  rstag:
  rsinittag:
  snmptag: v9.0.040.07
  pullPolicy: Always
  imagePullSecrets:
    - name: "pureengage-docker-staging"

## liveness and readiness probes
## !!! THESE OPTIONS SHOULD NOT BE CHANGED UNLESS INSTRUCTED BY GENESYS !!!
livenessValues:
  path: /ems-rs/components
  initialDelaySeconds: 30
  periodSeconds: 120
  timeoutSeconds: 3
  failureThreshold: 3

readinessValues:
  path: /ems-rs/components
  initialDelaySeconds: 10
```

```
  periodSeconds: 60
  timeoutSeconds: 3
  failureThreshold: 3

## PVCs defined
volumes:
  pvc:
    storageClass: managed-premium
    claimSize: 20Gi
    activemqAndLocalConfigPath: "/billing/gvp-rs"

## Define service(s) for application.  Fields many need to be modified based on `type`
service:
  type: ClusterIP
  restapiport: 8080
  activemqport: 61616
  envinjectport: 443
  dnsport: 53
  configserverport: 8888
  snmpport: 1705

## ConfigMaps with Configuration
## Use Config Map for creating environment variables
context:
  env:
    CFGAPP: default
    GVP_RS_SERVICE_HOSTNAME: gvp-rs.gvp.svc.cluster.local
    #CFGPASSWORD: password
    #CFGUSER: default
    CFG_HOST: gvp-configserver.gvp.svc.cluster.local
    CFG_PORT: '8888'
    CMDLINE: ./rs_startup.sh
    DBNAME: gvp_rs
    #DBPASS: 'jbIKfoS6LpfgaU$E'
    DBUSER: openshiftadmin
    rsDbSharedUsername: openshiftadmin
    DBPORT: 1433
    ENVTYPE: staging
    GenesysIURegion: westus2
    localconfigcachepath: /billing/gvp-rs/data/cache
    HOSTFOLDER: Hosts
    HOSTOS: CFGRedHatLinux
    LCAPORT: '4999'
    MSSQLHOST: mssqlserveropenshift.database.windows.net
    RSAPP: azure_rs
    RSJVM_INITIALHEAPSIZE: 500m
    RSJVM_MAXHEAPSIZE: 1536m
    RSFOLDER: Applications
    RS_VERSION: 9.0.032.22
    STDOUT: 'true'
    WRKDIR: /usr/local/genesys/rs/
    SNMPAPP: azure_rs_snmp
    SNMP_WORKDIR: /usr/sbin
    SNMP_CMDLINE: snmpd
    SNMPFOLDER: Applications

  RSCONFIG:
    messaging:
      activemq.memoryUsageLimit: "256 mb"
      activemq.dataDirectory: "/billing/gvp-rs/data/activemq"
    log:
      verbose: "trace"
      trace: "stdout"
```

```
    dbmp:
      rs.db.retention.operations.daily.default: "40"
      rs.db.retention.operations.monthly.default: "40"
      rs.db.retention.operations.weekly.default: "40"
      rs.db.retention.var.daily.default: "40"
      rs.db.retention.var.monthly.default: "40"
      rs.db.retention.var.weekly.default: "40"
      rs.db.retention.cdr.default: "40"

# Default secrets storage to k8s secrets with csi able to be optional
secret:
  # keyVaultSecret will be a flag to between secret types(k8's or CSI). If keyVaultSecret was
set to false k8's secret will be used
  keyVaultSecret: false
  #RS SQL server secret
  rsSecretName: shared-gvp-rs-sqlserver-secret
  # secretProviderClassName will not be used used when keyVaultSecret set to false
  secretProviderClassName: keyvault-gvp-rs-sqlserver-secret-00
  dbreadersecretFileName: db-reader-password
  dbadminsecretFileName: db-admin-password
  #Configserver secret
  #If keyVaultSecret set to false the below parameters will not be used.
  configserverProviderClassName: gvp-configserver-secret
  cfgSecretFileNameForCfgUsername: configserver-username
  cfgSecretFileNameForCfgPassword: configserver-password
  #If keyVaultSecret set to true the below parameters will not be used.
  cfgServerSecretName: configserver-secret
  cfgSecretKeyNameForCfgUsername: username
  cfgSecretKeyNameForCfgPassword: password

## Ingress configuration
ingress:
  enabled: false
  annotations: {}
    # kubernetes.io/ingress.class: nginx
  # kubernetes.io/tls-acme: "true"
  hosts:
    - host: chart-example.local
      paths: []
  tls: []
  #  - secretName: chart-example-tls
  #    hosts:
  #      - chart-example.local

networkPolicies:
  enabled: false

## primaryAppresource requests and limits
## ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resourceForRS:
  # We usually recommend not to specify default resources and to leave this as a conscious
  # choice for the user. This also increases chances charts run on environments with little
  # resources, such as Minikube. If you do want to specify resources, uncomment the following
  # lines, adjust them as necessary, and remove the curly braces after 'resources:'.
  requests:
    memory: "500Mi"
    cpu: "200m"
  limits:
    memory: "1Gi"
    cpu: "300m"

resoueceForSnmp:
```

```
  requests:
    memory: "500Mi"
    cpu: "100m"
  limits:
    memory: "1Gi"
    cpu: "150m"

## primaryApp containers' Security Context
## ref: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/#set-the-
security-context-for-a-container
##
## Containers should run as genesys user and cannot use elevated permissions
securityContext:
  runAsNonRoot: true
  runAsUser: 500
  runAsGroup: 500

podSecurityContext:
  fsGroup: 500

## Priority Class
## ref: https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption/
##
priorityClassName: ""

## Affinity for assignment.
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-
affinity
##
affinity: {}

## Node labels for assignment.
## ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Tolerations for assignment.
## ref: https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/
##
tolerations: []

## Extra labels
## ref: https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/
##
#  labels: {}

## Extra Annotations
## ref: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/
##
#  annotations: {}

## Service/Pod Monitoring Settings
monitoring:
  podMonitorEnabled: true
  prometheusRulesEnabled: true
  grafanaEnabled: true

monitor:
  prometheusPort: 9116
  monitorName: gvp-monitoring
  module: [if_mib]
  target: [127.0.0.1:1161]
```

```
##DNS Settings
dnsConfig:
  options:
    - name: ndots
      value: "3"
```

Verify the deployed resources

Verify the deployed resources from the OpenShift console/CLI.

4. GVP Resource Manager

Note: Resource Manager and forward will not pass readiness checks until an MCP has registered properly. This is because this service is not available without MCPs.

Persistent Volumes creation

Create the following PVs that are required for the service deployment.

Note: If your OpenShift deployment is capable of self-provisioning of Persistent Volumes, you can skip this step. The provisioner will create the volumes.

gvp-rm-01

gvp-rm-01-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-rm-01
spec:
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: gvp
  nfs:
    path:  /export/vol1/PAT/gvp/rm-01
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-rm-01-pv.yaml
```

gvp-rm-02

gvp-rm-02-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-rm-02
spec:
```

```
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: gvp
  nfs:
    path:  /export/vol1/PAT/gvp/rm-02
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-rm-02-pv.yaml
```

gvp-rm-logs-01

gvp-rm-logs-01-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-rm-logs-01
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: gvp
  nfs:
    path:  /export/vol1/PAT/gvp/rm-logs-01
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-rm-logs-01-pv.yaml
```

gvp-rm-logs-02

gvp-rm-logs-02-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-rm-logs-02
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: gvp
  nfs:
    path:  /export/vol1/PAT/gvp/rm-logs-02
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-rm-logs-02-pv.yaml
```

Install Helm chart

Download the required Helm chart release from the JFrog repository and install. Refer to Helm Chart URLs.

```
helm install gvp-rm ./ -f gvp-rm-values.yaml
```

Set the following values in your values.yaml for Configuration Server:

- priorityClassName >> Set to a priority class that exists on the cluster (or create it instead).
- imagePullSecrets >> Set to your pull secret name.
- Set cfgServerSecretName if you changed it from default.

gvp-rm-values.yaml

```
## Global Parameters
## Add labels to all the deployed resources
##
labels:
  enabled: true
  serviceGroup: "gvp"
  componentType: "shared"

## Primary App Configuration
##
# primaryApp:
# type: ReplicaSet
# Should include the defaults for replicas
deployment:
  replicaCount: 2
  deploymentEnv: "UPDATE_ENV"
  namespace: gvp
  clusterDomain: "svc.cluster.local"
nameOverride: ""
fullnameOverride: ""

image:
  registry: pureengage-docker-staging.jfrog.io
  gvprmrepository: gvp/gvp_rm
  cfghandlerrepository: gvp/gvp_rm_cfghandler
  snmprepository: gvp/gvp_snmp
  gvprmtestrepository: gvp/gvp_rm_test
  cfghandlertag:
  rmtesttag:
  rmtag:
  snmptag: v9.0.040.07
  pullPolicy: Always
  imagePullSecrets:
    - name: "pureengage-docker-staging"

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

# Pod termination grace period 15 mins.
gracePeriodSeconds: 900

## liveness and readiness probes
```

```
## !!! THESE OPTIONS SHOULD NOT BE CHANGED UNLESS INSTRUCTED BY GENESYS !!!
livenessValues:
  path: /rm/liveness
  initialDelaySeconds: 60
  periodSeconds: 90
  timeoutSeconds: 20
  failureThreshold: 3

readinessValues:
  path: /rm/readiness
  initialDelaySeconds: 10
  periodSeconds: 60
  timeoutSeconds: 20
  failureThreshold: 3

## PVCs defined
volumes:
  billingpvc:
    storageClass: managed-premium
    claimSize: 20Gi
    mountPath: "/rm"
  logpvc:
    EnablePVForLogStorage: true
    storageClass: managed-premium
    claimSize: 5Gi
    accessMode: ReadWriteOnce
    mountPath: "/mnt/log"
    # If PV is not used for log storage by disabling the flag EnablePVForLogStorage: false,
the given host path will be used for log storage.
    LogStorageHostPath: /mnt/log

## Define service(s) for application.  Fields many need to be modified based on `type`
service:
  type: ClusterIP
  port: 5060
  rmHealthCheckAPIPort: 8300

## ConfigMaps with Configuration
## Use Config Map for creating environment variables
context:
  env:
    cfghandler:
      CFGSERVER: gvp-configserver.gvp.svc.cluster.local
      CFGSERVERBACKUP: gvp-configserver.gvp.svc.cluster.local
      CFGPORT: "8888"
      CFGAPP: "default"
      RMAPP: "azure_rm"
      RMFOLDER: "Applications\\RM_MicroService\\RM_Apps"
      HOSTFOLDER: "Hosts\\RM_MicroService"
      MCPFOLDER: "MCP_Configuration_Unit\\MCP_LRG"
      SNMPFOLDER: "Applications\\RM_MicroService\\SNMP_Apps"
      EnvironmentType: "prod"
      CONFSERVERAPP: "confserv"
      RSAPP: "azure_rs"
      SNMPAPP: "azure_rm_snmp"
      STDOUT: "true"
      VOICEMAILSERVICEDIDNUMBER: "55551111"

  RMCONFIG:
    rm:
      sip-header-for-dnis: "Request-Uri"
      ignore-gw-lrg-configuration: "true"
      ignore-ruri-tenant-dbid: "true"
```

```
    log:
      verbose: "trace"
    subscription:
      sip.transport.dnsharouting: "true"
      sip.headerutf8verification: "false"
      sip.transport.setuptimer.tcp: "5000"
      sip.threadpoolsize: "1"
    registrar:
      sip.transport.dnsharouting: "true"
      sip.headerutf8verification: "false"
      sip.transport.setuptimer.tcp: "5000"
      sip.threadpoolsize: "1"
    proxy:
      sip.transport.dnsharouting: "true"
      sip.headerutf8verification: "false"
      sip.transport.setuptimer.tcp: "5000"
      sip.threadpoolsize: "16"
      sip.maxtcpconnections: "1000"
    monitor:
      sip.transport.dnsharouting: "true"
      sip.maxtcpconnections: "1000"
      sip.headerutf8verification: "false"
      sip.transport.setuptimer.tcp: "5000"
      sip.threadpoolsize: "1"
    ems:
      rc.cdr.local_queue_path: "/rm/ems/data/cdrQueue_rm.db"
      rc.ors.local_queue_path: "/rm/ems/data/orsQueue_rm.db"

# Default secrets storage to k8s secrets with csi able to be optional
secret:
  # keyVaultSecret will be a flag to between secret types(k8's or CSI). If keyVaultSecret was
set to false k8's secret will be used
  keyVaultSecret: false
  #If keyVaultSecret set to false the below parameters will not be used.
  configserverProviderClassName: gvp-configserver-secret
  cfgSecretFileNameForCfgUsername: configserver-username
  cfgSecretFileNameForCfgPassword: configserver-password
  #If keyVaultSecret set to true the below parameters will not be used.
  cfgServerSecretName: configserver-secret
  cfgSecretKeyNameForCfgUsername: username
  cfgSecretKeyNameForCfgPassword: password

## Ingress configuration
ingress:
  enabled: false
  annotations: {}
    # kubernetes.io/ingress.class: nginx
  # kubernetes.io/tls-acme: "true"
  paths: []
  hosts:
    - chart-example.local
  tls: []
  #  - secretName: chart-example-tls
  #    hosts:
  #      - chart-example.local
networkPolicies:
  enabled: false
sip:
  serviceName: sipnode

## primaryAppresource requests and limits
## ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
```

```
resourceForRM:
  # We usually recommend not to specify default resources and to leave this as a conscious
  # choice for the user. This also increases chances charts run on environments with little
  # resources, such as Minikube. If you do want to specify resources, uncomment the following
  # lines, adjust them as necessary, and remove the curly braces after 'resources:'.
  requests:
    memory: "1Gi"
    cpu: "200m"
    ephemeral-storage: "10Gi"
  limits:
    memory: "2Gi"
    cpu: "250m"

resoueceForSnmp:
  requests:
    memory: "500Mi"
    cpu: "100m"
  limits:
    memory: "1Gi"
    cpu: "150m"

## primaryApp containers' Security Context
## ref: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/#set-the-
security-context-for-a-container
##
## Containers should run as genesys user and cannot use elevated permissions
securityContext:
  fsGroup: 500
  runAsNonRoot: true
  runAsUserRM: 500
  runAsGroupRM: 500
  runAsUserCfghandler: 500
  runAsGroupCfghandler: 500

## Priority Class
## ref: https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption/
##
priorityClassName: ""

## Affinity for assignment.
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-
affinity
##
affinity: {}

## Node labels for assignment.
## ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector:


## Tolerations for assignment.
## ref: https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/
##
tolerations: []

## Service/Pod Monitoring Settings
monitoring:
  podMonitorEnabled: true
  prometheusRulesEnabled: true
  grafanaEnabled: true

monitor:
```

```
  monitorName: gvp-monitoring
  prometheusPort: 9116
  prometheusPortlogs: 8200
  logFilePrefixName: RM
  module: [if_mib]
  target: [127.0.0.1:1161]
```

Verify the deployed resources

Verify the deployed resources from the OpenShift console/CLI.

5. GVP Media Control Platform

Persistent Volumes creation

Create the following PVs that are required for the service deployment.

gvp-mcp-logs-01

gvp-mcp-logs-01-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-mcp-logs-01
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: gvp
  nfs:
    path:  /export/vol1/PAT/gvp/mcp-logs-01
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-mcp-logs-01-pv.yaml
```

gvp-mcp-logs-02

gvp-mcp-logs-02-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-mcp-logs-02
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: gvp
```

```
  nfs:
    path:  /export/vol1/PAT/gvp/mcp-logs-02
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-mcp-logs-02-pv.yaml
```

gvp-mcp-rup-volume-01

gvp-mcp-rup-volume-01-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-mcp-rup-volume-01
spec:
  capacity:
    storage: 40Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: disk-premium
  nfs:
    path:  /export/vol1/PAT/gvp/mcp-logs-01
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-mcp-rup-volume-01-pv.yaml
```

gvp-mcp-rup-volume-02

gvp-mcp-rup-volume-02-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-mcp-rup-volume-02
spec:
  capacity:
    storage: 40Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: disk-premium
  nfs:
    path:  /export/vol1/PAT/gvp/mcp-logs-02
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-mcp-rup-volume-02-pv.yaml
```

gvp-mcp-recording-volume-01

gvp-mcp-recordings-volume-01-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
```

```
metadata:
  name: gvp-mcp-recording-volume-01
spec:
  capacity:
    storage: 40Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: gvp
  nfs:
    path:  /export/vol1/PAT/gvp/mcp-logs-01
    server: 192.168.30.51
```

Run the following command:

```
kubectl create -f gvp-mcp-recordings-volume-01-pv.yaml
```

gvp-mcp-recording-volume-02

gvp-mcp-recordings-volume-02-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gvp-mcp-recording-volume-02
spec:
  capacity:
    storage: 40Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: gvp
  nfs:
    path:  /export/vol1/PAT/gvp/mcp-logs-02
    server: 192.168.30.51
```

Run the following command:

```
kubecl create -f gvp-mcp-recordings-volume-02-pv.yaml
```

Install Helm chart

Download the required Helm chart release from the JFrog repository and install. Refer to Helm Chart URLs.

```
helm install gvp-mcp ./ -f gvp-mcp-values.yaml
```

Set the following values in your values.yaml:

- Set **logicalResourceGroup: "MCP_Configuration_Unit"** to add MCPs to the Real Configuration Unit (rather than test).

gvp-mcp-values.yaml

```
## Default values for gvp-mcp.
## This is a YAML-formatted file.
## Declare variables to be passed into your templates.

## Global Parameters
```

```
## Add labels to all the deployed resources
##
podLabels: {}

## Add annotations to all the deployed resources
##
podAnnotations: {}

serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name:

## Deployment Configuration
deploymentEnv: "UPDATE_ENV"
replicaCount: 2
terminationGracePeriod: 3600

## Name and dashboard overrides
nameOverride: ""
fullnameOverride: ""
dashboardReplicaStatefulsetFilterOverride: ""

## Base Labels. Please do not change these.
serviceName: gvp-mcp
component: shared
partOf: gvp

## Command-line arguments to the MCP process
args:
  - "gvp-configserver"
  - "8888"
  - "default"
  - "/etc/mcpconfig/config.ini"

## Container image repo settings.
image:
  mcp:
    registry: pureengage-docker-staging.jfrog.io
    repository: gvp/multicloud/gvp_mcp
    tag: "{{ .Chart.AppVersion }}"
    pullPolicy: IfNotPresent
  serviceHandler:
    registry: pureengage-docker-staging.jfrog.io
    repository: gvp/multicloud/gvp_mcp_servicehandler
    tag: "{{ .Chart.AppVersion }}"
    pullPolicy: IfNotPresent
  configHandler:
    registry: pureengage-docker-staging.jfrog.io
    repository: gvp/multicloud/gvp_mcp_confighandler
    tag: "{{ .Chart.AppVersion }}"
    pullPolicy: IfNotPresent
  snmp:
    registry: pureengage-docker-staging.jfrog.io
    repository: gvp/multicloud/gvp_snmp
    tag: v9.0.040.21
    pullPolicy: IfNotPresent
  rup:
    registry: pureengage-docker-staging.jfrog.io
```

```
    repository: cce/recording-provider
    tag: 9.0.000.00.b.1432.r.ef30441
    pullPolicy: IfNotPresent

## MCP specific settings
mcp:
  ## Settings for liveness and readiness probes of MCP
  ## !!! THESE VALUES SHOULD NOT BE CHANGED UNLESS INSTRUCTED BY GENESYS !!!
  livenessValues:
    path: /mcp/liveness
    initialDelaySeconds: 30
    periodSeconds: 60
    timeoutSeconds: 20
    failureThreshold: 3
    healthCheckAPIPort: 8300

  # Used instead of startupProbe.  This runs all initial self-tests, and could take some time.
  # Timeout is = 2
# maxUnavailable = 1
hpa:
  enabled: false
  minReplicas: 2
  maxUnavailable: 1
  maxReplicas: 4
  podManagementPolicy: Parallel
  targetCPUAverageUtilization: 20
  scaleupPeriod: 15
  scaleupPods: 4
  scaleupPercent: 50
  scaleupStabilizationWindow: 0
  scaleupPolicy: Max
  scaledownPeriod: 300
  scaledownPods: 2
  scaledownPercent: 10
  scaledownStabilizationWindow: 3600
  scaledownPolicy: Min

## Service/Pod Monitoring Settings
prometheus:
  mcp:
    name: gvp-mcp-snmp
    port: 9116

  rup:
    name: gvp-mcp-rup
    port: 8080

  podMonitor:
    enabled: true

grafana:
  enabled: false

  #log:
  #  name: gvp-mcp-log
  #  port: 8200

## Pod Disruption Budget Settings
podDisruptionBudget:
  enabled: true

## Enable network policies or not
networkPolicies:
```

```
  enabled: false

## DNS configuration options
dnsConfig:
  options:
    - name: ndots
      value: "3"

## Configuration overrides
mcpConfig:
  # MCP config overrides
  mcp.mpc.numdispatchthreads: 4
  mcp.log.verbose: "interaction"
  mcp.mpc.codec: "pcmu pcma telephone-event"
  mcp.mpc.transcoders: "PCM MP3"
  mcp.mpc.playcache.enable: 1
  mcp.fm.http_proxy: ""
  mcp.fm.https_proxy: ""

  #MRCP v2 ASR config overrides
  mrcpv2_asr.provision.vrm.client.connectpersetup: true
  mrcpv2_asr.provision.vrm.client.disablehotword: false
  mrcpv2_asr.provision.vrm.client.hotkeybasepath: "/usr/local/genesys/mcp/grammar/nuance/
hotkey"
  mrcpv2_asr.provision.vrm.client.noduplicatedgramuri: true
  mrcpv2_asr.provision.vrm.client.sendswmsparams: false
  mrcpv2_asr.provision.vrm.client.transportprotocol: "MRCPv2"
  mrcpv2_asr.provision.vrm.client.sendloggingtag: true
  mrcpv2_asr.provision.vrm.client.resource.name: "NuanceASRv2"
  mrcpv2_asr.provision.vrm.client.resource.uri: "sip:mresources@speech-server-clusterip:5060"
  mrcpv2_asr.provision.vrm.client.tlscertificatekey: "/usr/local/genesys/mcp/config/
x509_certificate.pem"
  mrcpv2_asr.provision.vrm.client.tlsprivatekey: "/usr/local/genesys/mcp/config/
x509_certificate.pem"
  mrcpv2_asr.provision.vrm.client.tlspassword: ""
  mrcpv2_asr.provision.vrm.client.tlsprotocoltype: "TLSv1"
  mrcpv2_asr.provision.vrm.client.confidencescale: 1
  mrcpv2_asr.provision.vrm.client.sendsessionxml: true
  mrcpv2_asr.provision.vrm.client.supportfornuance11: true
  mrcpv2_asr.provision.vrm.client.uniquegramid: true

  #MRCP v2 TTS config overrides
  mrcpv2_tts.provision.vrm.client.connectpersetup: true
  mrcpv2_tts.provision.vrm.client.speechmarkerencoding: "UTF-8"
  mrcpv2_tts.provision.vrm.client.transportprotocol: "MRCPv2"
  mrcpv2_tts.provision.vrm.client.sendloggingtag: true
  mrcpv2_tts.provision.vrm.client.resource.name: "NuanceTTSv2"
  mrcpv2_tts.provision.vrm.client.resource.uri: "sip:mresources@speech-server-clusterip:5060"
  mrcpv2_tts.provision.vrm.client.tlscertificatekey: "/usr/local/genesys/mcp/config/
x509_certificate.pem"
  mrcpv2_tts.provision.vrm.client.tlsprivatekey: "/usr/local/genesys/mcp/config/
x509_certificate.pem"
  mrcpv2_tts.provision.vrm.client.tlspassword: ""
  mrcpv2_tts.provision.vrm.client.tlsprotocoltype: "TLSv1"
  mrcpv2_tts.provision.vrm.client.nospeechlanguageheader: true
  mrcpv2_tts.provision.vrm.client.sendsessionxml: true
  mrcpv2_tts.provision.vrm.client.supportfornuance11: true
```

Verify the deployed resources

Verify the deployed resources from Google console/CLI.