Outbound (CX Contact) Private Edition Guide

Deploy CX Contact

2/7/2025

Contents

- 1 Assumptions
- 2 Prepare cluster resources
  - 2.1 Create the Storage Class
- 3 Deploy CX Contact
- 4 Validate the deployment
- 5 Configure monitoring and logging

Learn how to deploy CX Contact 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**
>
> Review Before you begin for the full list of prerequisites required to deploy CX Contact.

CX Contact is a shared service and is deployed in each region, as required. After deployment, it will be fully functional only in the tenant's primary region.

Prepare cluster resources

To prepare your cluster resources, create a storage class.

> **Tip**
> Creating the storage class is optional. If you have an existing storage class, you can use it and a different provisioner, but the storage class must have ReadWriteMany (RWX) capabilities.

Create the Storage Class

1. Log in to the cluster using the Command Line Interface (CLI).
2. Go to **Storage**, and click **Storage Claim** > **Create Storage Class**.
3. Click **Edit YAML**.
4. Update the values in the template using the example below that works with the setup of your environment.
   - If your cluster is on-premises (NFS-based storage):

     ```
     apiVersion: storage.k8s.io/v1
     kind: StorageClass
     metadata:
       name: cxc-storage
     provisioner: cluster.local/nfs-vce-c00ds-vol1-nfs-subdir-external-provisioner
     reclaimPolicy: Delete
     volumeBindingMode: Immediate
     ```

   - If your cluster is in the cloud (Azure files-based storage):

     ```
     apiVersion: storage.k8s.io/v1
     kind: StorageClass
     metadata:
       name: cxc-storage
     provisioner: kubernetes.io/azure-file
     reclaimPolicy: Delete
     volumeBindingMode: Immediate
     ```

5. Click **Create**.

Deploy CX Contact

Complete the following procedure using Kubectl and the Helm tools:

1. Create a new project or a namespace.
   - For Kubernetes, enter

     ```
     kubectl create namespace cxc
     ```

   - For GKE, enter

     ```
     kubectl create ns cxc
     ```

   - For AKS, enter

```
kubectl create namespace cxc
```

2. Create the pull secret.Use the following code snippets as examples of how to create the default pull secret:
   - For generic Kubernetes:

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

   - For GKE :

     ```
     kubectl create secret docker-registry mycred --docker-server= --docker-username= --docker-password= --docker-email=
     ```

   - For AKS:

     ```
     kubectl create secret docker-registry mycred --docker-server= --docker-username= --docker-password= --docker-email=
     ```

3. Download latest version of the CX Contact installation Helm Charts from the artifactory. See the JFrog Platform Artifactory.
4. Extract parameters from chart to see multiple (default) values used to fine-tune the installation.

   ```
   helm install cxc ./cxcontact-.tgz > values.yaml
   ```

   You can apply multiple override values to customize your setup. However, Genesys recommends using minimal overriding values in the installation:
   For example, override_values.yaml

   ```
   configserver:
     user_name: cloudcon
     user_password: cloudcon

   cxcontact:

     replicas: 2
     log:
       level: info

     compliance_data:
       cdp_url: false
       cdp_ng:
         gcloud_id: false
         gcloud_secret: false

   #  override:                #if connecting to Nexus. Otherwise Dial Manager is off
   #    dial-manager:
   #      enabled: false
   #      nexus:
   #        host: ..
   #        port: ..
   #      api_key: ..          #required

   monitoring:
     enabled: true
     dashboards: false
     alarms: false
     pagerduty: false


   redis:
   ```

```
  enabled: true
  cluster: true
  # can be comma-delimited list of redis nodes, for e.g.
  # nodes: redis://redis-node1:6379,redis://redis-node2:6379,redis://redis-node3:6379
  nodes: redis://infra-redis-redis-cluster.infra.svc.cluster.local:6379
  #use_tls: false
  requirepass: true
  password:

elasticsearch:
  enabled: true
  host: http://elastic-es-http.infra.svc.cluster.local
  port: 9200

gws:
  # secret in plain text
  client_id: cx_contact
  client_secret: cx_contact

  # GWS Ingress URL
  frontend_host: https://gauth.apps.
  frontend_port: 443


#  Services. Will be used for connection to GWS if GWS Internal Ingress URL is disabled
(empty values)
  core:
    auth:
      host: http://gauth-auth.gauth
      port: 80
    environment:
      host: http://gauth-environment.gauth
      port: 80
  platform:
    ocs:
      host: http://gws-service-proxy.gws
      port: 80
    configuration:
      host: http://gws-service-proxy.gws
      port: 80
    statistics:
      host: http://gws-service-proxy.gws
      port: 80
    setting:
      host: http://gws-service-proxy.gws
      port: 80
    voice:
      host: http://gws-service-proxy.gws
      port: 80


ingress:
  enabled: true
  #tls_enabled: false
  cxc_frontend: cxc.apps.
  annotations:
#    !!!Ingress Session Stickiness is required.
#    Default annotations:
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/session-cookie-samesite: "Lax"
    nginx.ingress.kubernetes.io/session-cookie-name: "cxc-session-cookie"
    nginx.ingress.kubernetes.io/proxy-body-size: "0"
  tls: []
```

```
  #  - hosts:
  #      - chart-example.local
  #    secretName: chart-example-tls

# Additional ingress to expose internal backend endpoints.
# If disabled  - all endpoints will be exposed on ingress.cxc_frontend
internal_ingress:
  enabled: true
  tls_enabled: false
  cxc_backend: cxc-int.apps.
  annotations:
#    Default annotations:
    nginx.ingress.kubernetes.io/proxy-body-size: "0"
    nginx.ingress.kubernetes.io/ssl-redirect: 'false'
  tls: []
  #  - hosts:
  #      - chart-example.local
  #    secretName: chart-example-tls

storage:
  # Persistent Volumes Claim Configuration
  pvc:
    enabled: true
    create: true
#    Instructs Helm to skip deleting PVC when a helm operation (such as helm uninstall,
helm upgrade or helm rollback)
#    would result in its deletion. However, this resource becomes orphaned. Helm will no
longer manage it in any way.
#    https://helm.sh/docs/howto/charts_tips_and_tricks/#tell-helm-not-to-uninstall-a-
resource
#    If PVC is already orphaned and you want to re-use it - set `storage.pvc.create` to
`false`.
    keepAfterDeletion: false
    size: 10Gi
    name: cxc-claim
    storageClassName: cxc-storage
```

5. Validate the Helm chart and provided values, enter:

```
$ helm template cxc ./cxcontact-.tgz -f override_values.yaml
```

6. Install the CX Contact chart, using the override values file that you prepared in step, enter:

```
$ helm install cxc ./cxcontact-.tgz -f override_values.yaml
```

7. If errors occur, verify the input values, YAML files syntax, and your Kubernetes context. enter:

```
$ kubectl config get-contexts
$ kubectl get pods -n cxc
$ kubectl describe pod  -n cxc
```

8. If troubleshooting is necessary, try adding the **--dry-run** command line parameter in **helm install ..** for verbose error output.

Tip

- To see the full set of available parameters, extract the default helm values from helm package:

```
$ helm show values cxcontact-.tgz > values.yaml
```

- For persistent volume claims in production, Genesys recommends 100-200 GB.

This completes the CX Contact shared service installation.

Next steps:

- Provision tenant for CX Contact/outbound. See Provision CX Contact.
- Validate the deployment.

Validate the deployment

1. Watch the helm output at the end of installation. It provides the status and additional information about where to log in to the CX Contact UI. See the following sample output:

```
Release "cxc" has been upgraded. Happy Helming!
NAME: cxc
LAST DEPLOYED: Tue Jul 13 10:18:07 2021
NAMESPACE: cxc
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Please be patient while CXContact  is being deployed


ENDPOINTS:
UI: http://cxc.apps./ui/cxcontact/#!/campaign/list
API basepath: http://cxc.apps./cx-contact/v3/
API swagger doc: http://cxc.apps./cx-contact/v3/explore
CXC backend basepath: http://cxc-int.apps./
```

2. Check the status of the Events and Pods. Pods should be up and running—8 CX Contact components in total (7, if you have not defined Nexus for digital outbound).
3. Check the **amark-app** and other pods logs, primarily for errors related to connectivity to Redis and Elasticsearch.
4. If you have provisioned tenants for CX Contact, log in to the CX Contact UI (use URL from the Helm output above) using the tenant's administrator credentials. If you can log in successfully, that confirms that CX Contact works with the Redis cluster:
   - Check the CX Contact **About** > **Versions** page for the health status of the CX Contact components. They should be all green.
   - Check the **Analytics** page. It should show your successful log in, and confirm that CX Contact works with Elasticsearch.
   - Try to create a test Contact list. If you succeed, CX Contact will display a success confirmation

message.

Configure monitoring and logging

CX Contact monitoring is enabled by default.

See monitoring details and how to configure logging parameters in Observability in Outbound (CX Contact).