



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Universal Contact Service Private Edition Guide

Deploy Universal Contact Service

7/27/2024

Contents

- 1 Assumptions
- 2 Create a project
 - 2.1 Google Kubernetes Engine
 - 2.2 Azure Kubernetes Service
- 3 Prepare the Helm values file
- 4 Deploy the service
- 5 Validate the deployment
 - 5.1 curl requests

Learn how to deploy Universal Contact Service (UCS) 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 UCS.

This chart bootstraps an UCS deployment on a Kubernetes cluster using the Helm package manager.

Create a project

Google Kubernetes Engine

1. Log in to the GKE cluster.

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

2. Create a new manifest for UCS.

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

3. Create the namespace by applying the manifest to the cluster.

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

4. Confirm if the namespace is created.

```
kubectl describe namespace ucsx
```

Azure Kubernetes Service

1. Log in to the AKS cluster.

```
az aks get-credentials --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER_NAME
```

2. Create a new manifest for UCS.

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

3. Create the namespace by applying the manifest to the cluster.

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

4. Confirm if the namespace is created.

```
kubectl describe namespace ucsx
```

Prepare the Helm values file

Create a file named **values-override.yaml** and set the following values depending on your environment.

```
# Default values for ucsx.
# This is a YAML-formatted file.
```

```
# Declare variables to be passed into your templates.

# * Deployment
# Only two possible values are supported: Deployment, ReplicaSet
deploymentType: Deployment

# * Replicacount
replicaCount: 1

# * Images
# Replace for your values: registry and secret
image:
  pullPolicy: IfNotPresent
  pullSecrets: [name: ${DOCKER_REGISTRY_SECRET_NAME}]
  registry: ${DOCKER_SERVER}
  repository: ucsx/ucsx

# * Pod configuration
affinity: {}

nodeSelector: {}

tolerations: []

priorityClassName: ''

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #     - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

podDisruptionBudget:
  enabled: false
  minAvailable: 1

podLabels: {}

podAnnotations: {}

hpa:
  enabled: false
  targetCPUPercent: 60
  minReplicas: 1
  maxReplicas: 10

resources:
  requests:
    memory: "500Mi"
    cpu: "300m"
  limits:
    memory: "1000Mi"
    cpu: "2000m"

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:

# * K8s secret and configmap
# If not set it will be created automatically
existingSecret: ucsx-secret
existingConfig: ucsx-config

# * Authentication
# Set your values.
gauth:
  auth:
    url: "http://${GAUTH_AUTH_URL_INTERNAL}:${GAUTH_AUTH_URL_PORT_INTERNAL}"
    clientId: "${GAUTH_CLIENT_ID}"
    clientSecret: "${GAUTH_CLIENT_SECRET}"
  env:
    url: "http://${GAUTH_ENV_URL_INTERNAL}:${GAUTH_ENV_URL_PORT_INTERNAL}"

# * ElasticSearch
# Replace with your values.
elasticsearch:
  url: "http://${ES_URL_INTERNAL}:${ES_PORT_INTERNAL}"

# * DB Parameters
# Set your values.
db:
  ssl: 'false'
  name: "${DB_NAME_SHARED}"
  host: "${DB_HOST_INTERNAL}"
  port: "${DB_PORT_INTERNAL}"
  user: "${DB_USER_SHARED}"
  password: "${DB_PASSWORD_SHARED}"

# * Service
service:
  enabled: true
  name: ucsx
  type: ClusterIP
  externalPort: ${UCSX_ENDPOINT_INTERNAL_EXTERNALPORT}
  env: {}
  annotations: {}

# * Ingress
ingress:
  enabled: true
  annotations: {}
  hosts:
    - host: "${UCSX_ENDPOINT}"
      paths:
        - path: '/ucs/v3/'
          port: ${UCSX_ENDPOINT_INTERNAL_EXTERNALPORT}
        - path: '/ucs/v3/config'
          port: ${UCSX_ENDPOINT_INTERNAL_CONFIGPORT}
  tls: []

# * Monitoring
monitoring:
  # Deploy ServiceMonitor
  enabled: false
  # Create PrometheusRule k8s object with alarm definitions
  alarms: false

```

```
# kibanaUrl:
# grafanaUrl:
# runbookUrl:

# * Network policies
# true or false
networkPolicies:
  enabled: false

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

# * Healthcheck (Liveness and Readiness)
livenessProbe:
  httpGet:
    path: /ucs/v3/healthcheck
    port: rest
  initialDelaySeconds: 5
  timeoutSeconds: 5
  periodSeconds: 30

readinessProbe:
  httpGet:
    path: /ucs/v3/healthcheck
    port: rest
  initialDelaySeconds: 10
  timeoutSeconds: 5
  periodSeconds: 30
```

Deploy the service

Install UCS X:

```
helm upgrade --install ucsx ucsxhelmrepo/ucsx --version= --namespace=ucsx --set
existingSecret='' --set existingConfig='' -f values-override.yaml
```

Validate the deployment

To validate the install:

```
kubectl get pods -n=ucsx -l "app.kubernetes.io/name=ucsx,app.kubernetes.io/instance=ucsx"
```

Verify that details of the UCS X service deployment information is displayed.

To check the logs:

```
kubectl get pods -n ucsx
kubectl logs
```

curl requests

To get a bearer token:

```
curl -X POST https:///auth/v3/oauth/  
token?grant_type=client_credentials&client_id=&client_secret=
```

To get a cluster version:

```
curl --request POST http:///ucs/v3/rest/request/get-version -H "Authorization: Bearer " -H  
'Content-Type: application/json'
```

To get **/master-dbschema**:

```
curl -X GET http:///ucs/v3/config/master-dbschema -H "Authorization: Bearer " -H 'Content-  
Type: application/json'
```