Intelligent Workload Distribution Private Edition Guide

6/4/2026

Table of Contents

Overview

About IWD 6

Architecture 8

High availability and disaster recovery 14

Configure and deploy

Before you begin 15

Configure IWD 19

Provision IWD 23

Deploy Intelligent Workload Distribution 28

Upgrade, roll back, or uninstall

Upgrade, roll back, or uninstall 36

Observability

Observability in Intelligent Workload Distribution 41

IWD metrics and alerts 44

Contents

- 1 Overview
- 2 Configure and deploy
- 3 Upgrade, roll back, or uninstall
- 4 Observability

Find links to all the topics in this guide.

Related documentation:

-
-
-

RSS:

- For private edition

Intelligent Workload Distribution (IWD) is a service available with the Genesys Multicloud CX private edition offering.

Overview

Learn more about IWD, its architecture, and how to support high availability and disaster recovery.

- About IWD
- Architecture
- High availability and disaster recovery

Configure and deploy

Find out how to configure and deploy IWD.

- Before you begin
- Configure IWD
- Provision IWD
- Deploy Intelligent Workload Distribution

Upgrade, roll back, or uninstall

Find out how to upgrade, roll back, or uninstall IWD.

- Upgrade, roll back, or uninstall

Observability

Learn how to monitor IWD with metrics and logging.

- IWD metrics and alerts

About IWD

Contents

- 1 Supported Kubernetes platforms

Learn about IWD and how it works in Genesys Multicloud CX private edition.

Related documentation:

-
-
-
-

RSS:

- For private edition

IWD takes work items from existing enterprise software applications (such as ERP, BPM, DCM, Salesforce) and homegrown systems, analyzes the business context of the work item—for example, the associated business process, product requested, or value of the customer making the request—and creates a Universal Queue, sorted on business value, that ensures that the most critical or highest-value work items are distributed to the right resource at the right time, regardless of media type, system or location.

With IWD, enterprises can effectively manage all customer service resources and business processes across the enterprise, going beyond the walls of the formal contact center and into other areas of the business like branch offices and experts in the back office.

Supported Kubernetes platforms

IWD service is supported on the following cloud platforms:

- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)

See the Intelligent Workload Distribution Release Notes for information about when support was introduced.

Architecture

Contents

- 1 Introduction
- 2 Architecture diagram — Connections
- 3 Connections table

Learn about Intelligent Workload Distribution architecture

Related documentation:

-
-
-
-

RSS:

- For private edition

Introduction

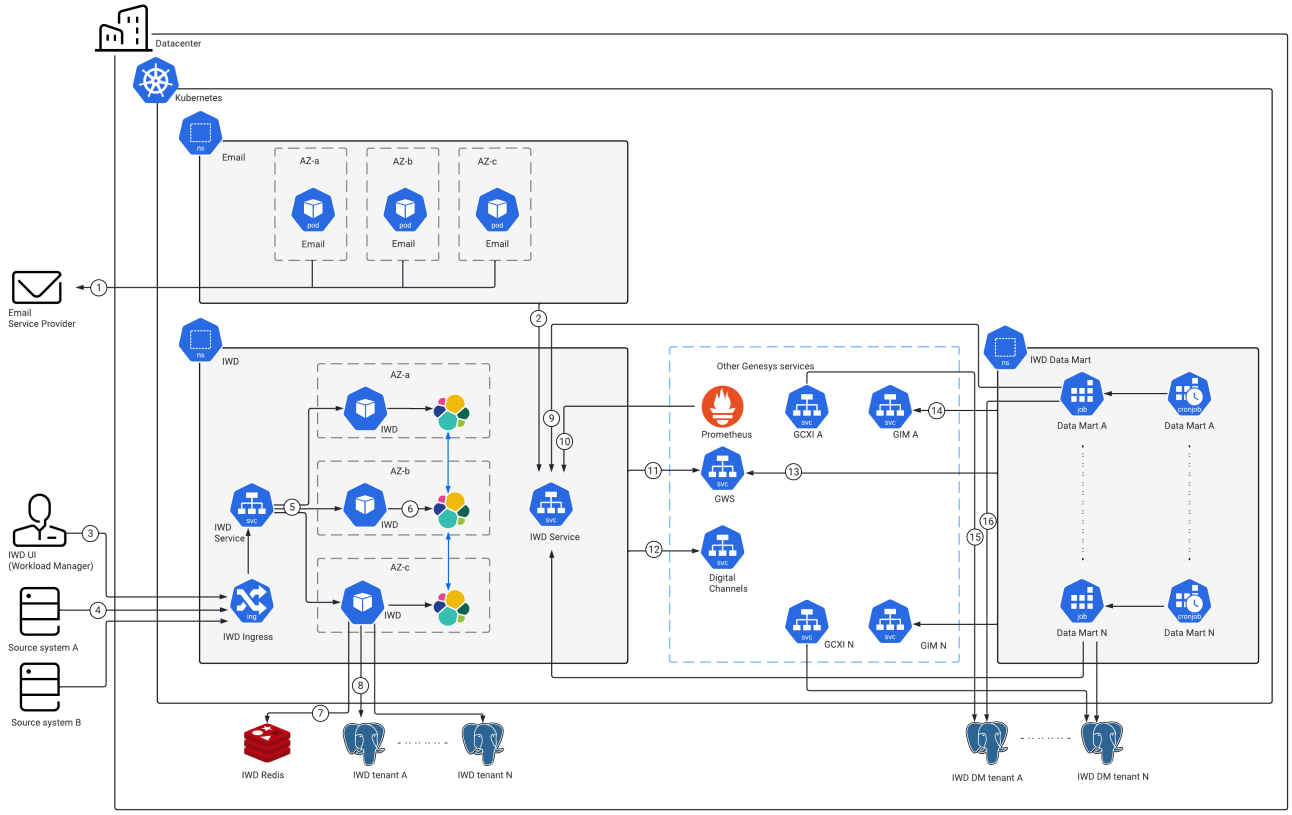
The architecture diagram in this topic illustrates a sample deployment of IWD, IWD Data Mart, and Email.

For information about the overall architecture of Genesys Multicloud CX private edition, see the high-level Architecture page.

See also High availability and disaster recovery for information about high availability/disaster recovery architecture.

Architecture diagram — Connections

The numbers on the connection lines refer to the connection numbers in the table that follows the diagram. The direction of the arrows indicates where the connection is initiated (the source) and where an initiated connection connects to (the destination), from the point of view of Intelligent Workload Distribution as a service in the network.



Connections table

The connection numbers refer to the numbers on the connection lines in the diagram. The **Source**, **Destination**, and **Connection Classification** columns in the table relate to the direction of the arrows in the Connections diagram above: The source is where the connection is initiated, and the destination is where an initiated connection connects to, from the point of view of Intelligent Workload Distribution as a service in the network. *Egress* means the Intelligent Workload Distribution service is the source, and *Ingress* means the Intelligent Workload Distribution service is the destination. *Intra-cluster* means the connection is between services in the cluster.

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
1	Email	Email Service Provider	HTTPS	25, 443, 587, 993	Egress	Email pulls email items from IMAP, Gmail, or Graph mailboxes. Pushes email items

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
						over SMTP, Graph, or Gmail.
2	Email	Intelligent Workload Distribution	TCP	4024	Egress	Email saves the following data in IWD: • Email items • Mailbox status • Mailbox configuration The port is configurable.
3	Source system	Application Gateway	HTTPS	443	Ingress	Data sync on work items.
4	Application Gateway	Intelligent Workload Distribution	TCP	4024	Ingress	Data sync on UI management, work items, configuration details, and reports.
5	IWD Service (Cluster IP)	IWD Pod	TCP	4024	Ingress	Data sync on UI management, work items, configuration details, and reports.
6	Intelligent Workload Distribution	Elasticsearch	TCP	9200	Egress	Data sync on work items.
7	Intelligent Workload Distribution	Redis	TCP	6379	Egress	Runtime cache. The port is configurable.
8	Intelligent Workload Distribution	PostgreSQL	TCP	5432	Egress	Data sync on work items, emails, and configuration details.
9	IWD Data Mart	Intelligent Workload	TCP	4024	Intra-cluster	Retrieves reporting

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
		Distribution				events from IWD.
10	Prometheus	Intelligent Workload Distribution	TCP	10052	Intra-cluster	Metrics for monitoring and alerting with Prometheus.
11	Intelligent Workload Distribution	Genesys Web Services and Applications	TCP	80	Intra-cluster	Task submission (for distribution), task retrieval, task stopping, and task status synchronization are done through GWS (pull).
12	Intelligent Workload Distribution	Digital Channels	TCP	80	Egress	IWD retrieves configuration data such as Tenant service configuration and API keys from Digital Channels.
13	IWD Data Mart	Genesys Web Services and Applications	HTTP	80	Intra-cluster	IWD Data Mart queries GWS to authorize and get token for requests to Digital Channels.
14	IWD Data Mart	Genesys Info Mart Postgres DB	TCP	5432	Egress	Agent details such as name are copied from GIM table to IWD DM table.
15	Genesys Customer Experience	IWD Datamart Tenant	TCP	5432	Intra-cluster	GCXI retrieves data

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
	Insights	Postgres DB				aggregated by IWD Data Mart to present it on the IWD related reports.
16	IWD Data Mart	IWD Datamart Tenant Postgres DB	TCP	5432	Egress	IWD Data Mart job saves the aggregated events.

High availability and disaster recovery

Find out how this service provides disaster recovery in the event the service goes down.

Related documentation:

-
-
-
-

RSS:

- For private edition

Service	High Availability	Disaster Recovery	Where can you host this service?
Intelligent Workload Distribution	N = 1 (singleton)	Not supported	Primary unit only

See High Availability information for all services: High availability and disaster recovery

IWD fails if either postgres or Redis fails, or becomes unavailable.

Before you begin

Contents

- 1 Limitations and assumptions
- 2 Download the Helm charts
- 3 Third-party prerequisites
- 4 Storage requirements
- 5 Network requirements
- 6 Browser requirements
- 7 Genesys dependencies
- 8 GDPR support

Find out what to do before deploying IWD.

Related documentation:

-
-
-
-

RSS:

- For private edition

Limitations and assumptions

The current version of IWD:

- supports single-region model of deployment only
- requires dedicated PostgreSQL deployment per customer

Download the Helm charts

IWD in Genesys Multicloud CX private edition includes the following containers:

- iwd

The service also includes a Helm chart, which you must deploy to install the required containers for IWD:

- iwd

See Helm Charts and Containers for IWD and IWD Data Mart for the Helm chart version you must download for your release.

To download the Helm chart, navigate to the **iwd** folder in the JFrog repository. For information about how to download the Helm charts, see Downloading your Genesys Multicloud CX containers.

Third-party prerequisites

Third-party services

Name	Version	Purpose	Notes
Elasticsearch	7.x	Used for text searching and indexing. Deployed per service that needs Elasticsearch during runtime.	Dedicated - one per deployment of IWD
Redis	6.x	Used for caching. Only distributions of Redis that support Redis cluster mode are supported, however, some services may not support cluster mode.	Dedicated - one per deployment of IWD
PostgreSQL	11.x	Relational database.	Dedicated instance for each tenant (recommended). In case of low load, one instance can host multiple DBs for multiple tenants (supported)
A container image registry and Helm chart repository		Used for downloading Genesys containers and Helm charts into the customer's repository to support a CI/CD pipeline. You can use any Docker OCI compliant registry.	
Load balancer		VPC ingress. For NGINX Ingress Controller, a single regional Google external network LB with a static IP and wildcard DNS entry will pass HTTPS traffic to NGINX Ingress Controller which will terminate SSL traffic and will be setup as part of the platform setup.	

Storage requirements

All data is stored in the PostgreSQL, Elasticsearch, and Digital Channels which are external to IWD.

Sizing of Elasticsearch depends on the load. Allow on average 15 KB per work item, 50 KB per email.

This can be adjusted depending on the size of items processed.

Network requirements

External Connections: IWD allows customer to configure webhooks. If configured, this establishes an HTTP or HTTPS connection to the configured host or port.

Browser requirements

Not applicable

Genesys dependencies

The following Genesys services are required:

- Genesys authentication service (GAuth)
- Universal Contact Service (UCS)
- Interaction Server
- Digital Channels (Nexus)

For the order in which the Genesys services must be deployed, refer to the Order of services deployment topic in the *Setting up Genesys Multicloud CX private edition* document.

GDPR support

Content coming soon

Configure IWD

Contents

- 1 Override Helm chart values
- 2 Configure Kubernetes
  - 2.1 ConfigMaps
  - 2.2 Create the pull secret

Learn how to configure IWD.

Related documentation:

-
-
-
-

RSS:

- For private edition

Override Helm chart values

The following table provides information on the IWD deployment settings that can be configured in the **values.yaml** file:

Parameter	Description	Default
`deploymentType`	Deployment type. Only two possible values are supported: Deployment, ReplicaSet	Deployment
`replicaCount`	Number of pods to be created	1
`image.registry`	Docker registry for iWD	pureengage-docker-staging.jfrog.io
`image.repository`	iWD Image name	nexus/iwd
`image.pullPolicy`	Image pull policy	IfNotPresent
`image.pullSecrets`	Specify docker-registry secret names as an array	[]
`affinity`	Map of node/pod affinities	{}
`nodeSelector`	Node labels for pod assignment	{}
`tolerations`	Tolerations for pod assignment	nil
`priorityClassName`	Priority class name	
`podSecurityContext`	Pod security context	{}
`securityContext`	Security context	{}
`podDisruptionBudget.enabled`	Enable or disable pod disruption budget	false
`podDisruptionBudget.minAvailable`	Set minimal number of pods available during the disruption	1
`podAnnotations`	Add annotations to pods	{}

Parameter	Description	Default
`podLabels`	Add custom labels to pods	{}
`hpa.enabled`	Enable or disable Horizontal Pod Autoscaler (HPA)	false
`hpa.minReplicas`	Minimal replicas count for HPA	1
`hpa.maxReplicas`	Maximal replicas count for HPA	10
`hpa.targetCPUPercent`	Specify target CPU utilization for HPA	60
`resources.limits.cpu`	Maximum amount of CPU K8s allocates for container	2000m
`resources.limits.memory`	Maximum amount of Memory K8s allocates for container	2000Mi
`resources.requests.cpu`	Guaranteed CPU allocation for container	300m
`resources.requests.memory`	Guaranteed Memory allocation for container	500Mi
`serviceAccount.create`	Specifies whether a service account should be created	false
`serviceAccount.annotations`	Annotations to add to service account	{}
`serviceAccount.name`	Service account name	""
`existingSecret`	Specify Secret name to read application secrets from	nil
`gauth.auth.url`	URL to Authentication service	nil
`gauth.auth.redirectUrl`	Redirect URL to Authentication service	nil
`redis.nodes`	Comma separate list of Redis nodes to connect	nil
`redis.useCluster`	Redis deployment mode	false
`redis.enableTLS`	Either to use TLS on Redis connection	false
`redis.password`	Access key for Redis authentication	nil
`nexus.url`	URL to Nexus	nil
`nexus.apikey`	Nexus API key	nil
`service.type`	Service type	ClusterIP
`monitoring.enabled`	Enable or disable pod monitor	false
`monitoring.alarms`	Create PrometheusRule k8s object with alarm definitions	false
`monitoring.dashboards`	Create ConfigMap with Grafana Dashboards	false
`networkPolicies.enabled`	Enable or disable network policies	false
`dnsConfig.options`	DNS Configuration options	{ name: ndots, value: "3" }

Configure Kubernetes

ConfigMaps

Not applicable as all required ConfigMaps are created via Helm Chart basing on the provided values.

Create the pull secret

Use the following code snippet as an example of how to create pull secret:

```
kubectl create secret docker-registry mycred --docker-server=pureengage.jfrog.io --docker-
username= --docker-password=
```

You can add *mycred* to Helm override values by setting **image.pullSecrets** to *[mycred]*.

Provision IWD

Contents

- 1 Provisioning via IWD API
  - 1.1 Create tenant request
  - 1.2 Parameters
  - 1.3 Delete tenant request
- 2 Manual provisioning

- Administrator

Learn how to provision IWD.

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
> Provisioning must be done after deploying IWD.

Workload Manager (UI) uses Roles. Agents must be assigned appropriate Roles.

All other provisioning is done through the tenant provisioning which can be performed manually or via IWD API.

Provisioning via IWD API

The following endpoints are available:

- POST /provisioning
- PUT /provisioning
- DELETE /provisioning/ccid

Currently, PUT and POST serve the same purpose.

Create tenant request

The following request assembles configurations for **GAPI**, **iWD**, and **iWDEmail** services, requested API keys provided in the **iwd.apiKeys** object, and provisions them in the Digital Channels database using Digital Channels API.

```
POST /iwd/v3/provisioning

BODY:

{
    "tenant": {
        "id": "100",   #tenant id - needed for correct generation of apikeys
        "name": "t100", #tenant Name
        "ccid": "9350e2fc-a1dd-4c65-",  #tenant CC id
        "apiKey": "049111b4-6cac-4e83-a7bf-37c057b45b0f", #IWD cluster API key from nexus
        "gws": {
            "gwsUrl": "http://gws-service-proxy.gws", #gws URL
            "data": {
                "authUrl": "http://gauth-auth.gauth", #gauth URL
                "authredircetUrl": "https://gauth.nlb01-uswest1.gcpe001.gencpe.net" #gws
redirect URL
            },
            "secret": {
                "clientId": "iwd_client", #gws API client id
                "clientSecret": "secret", #secret
                "apikey": "none",
                "username": "nexus", #admin user in configserver
                "token": ":;pYV1lNWh0-" #admin user password
            }
        }
    },
    "iwd": {
        "url": "http://iwd.iwd:4024",
        "db": {
                 #iwd DB details
            "host": "pgdb-dgt-postgresql.infra",
            "port": 5432,
            "database": "iwd-100",
            "user": "iwd-100",
            "password": "iwd-100",
            "ssl": false
        },
        "apiKeys": {
            "IWD_APIKEY_TENANT": "22552b96-8783-46a9-b0eb-075ddfa8893e", #New API key that
needs to be provisioned for iwd
            "IWD_APIKEY_IWDDM": "d6b68a7b-12ed-4c3c-830d-215eecdd1a48" #New API key that
needs to be provisioned for iwddm
        }
    },
    "iwdEmail": {
        "url": "N/A"
    }
}
```

Parameters

The following parameters are supported:

Parameter name	Datatype	Required	Description
tenant	object	true	Tenant to provision
tenant.name	string	true	Tenant name (GWS domain)
tenant.id	string	true	Short tenant ID

Parameter name	Datatype	Required	Description
tenant.ccid	string	true	GWS contact center ID
tenant.apikey	string	true	API key, issued by Digital Channels for a given tenant
iwd	object	true	IWD Service configuration
iwd.url	string	true	URL where IWD service is launched
iwd.db	object	true	IWD DB connection configuration
iwd.db.host	string	true	DB host
iwd.db.port	number	true	DB port
iwd.db.database	string	true	DB name
iwd.db.user	string	true	DB user
iwd.db.password	string	true	DB password
iwd.db.ssl	boolean	true	Use secure connection to DB or not
iwdEmail	object	true	IWD Email Service configuration
iwdEmail.url	string	true	URL where IWD Email service is launched, should be N/A
iwd.apiKeys	object	true	Tenant API keys for using various services
iwd.apiKeys.IWD_APIKEY_TENANT	string	true	Key for submitting API requests from customer
iwd.apiKeys.IWD_APIKEY_IWDDM	string	false	Required if you plan to deploy IWD Data Mart and use IWD reports

Response

The following success response is returned when the tenant is provisioned successfully:

```
{
   created: '''true'''
}
```

As a result, the following records are created in the Digital Channels database:

- GAPI service with GWS API key, GWS client secret, GWS username, and token
- IWD service with default options, categories, filters, prioritization, and secret
- Email service with the default options and mailboxes
- API keys

An error is returned if any of the required parameters is missing.

Delete tenant request

The following request removes the specified tenant provisioning:

```
DELETE /iwd/v3/provisioning/ccid?[service=String || deleteTenant=Boolean]
```

Path parameters

Parameter name	Datatype	Required	Description
ccid	string	true	Contact Center id

Query parameters

A minimum of one parameter is mandatory.

Parameter name	Datatype	Required	Description
service	string	true (at least one)	Specific services names to be deleted from the database
deleteTenant	boolean		Flag to delete all tenant services from Digital Channels database

Delete tenant examples:

To delete the specified services from the Digital Channels database:

```
DELETE /iwd/v3/provisioning/ccid?service=iWD,iWDEmail
```

To delete all tenant services and tenant configuration from the Digital Channels database:

```
DELETE /iwd/v3/provisioning/ccid?deleteTenant=true
```

Manual provisioning

Example:

```
curl 'http://iwd.iwd:4024/iwd/v3/provisioning' \

-H 'Content-Type: application/json; charset=utf-8' \

-H 'x-api-key: ed99c91d-dd18-4c96-af8e-86f8e8105bc4' \

--data '{"tenant":{"id":"100",
"name":"t100","ccid":"d846c51e-1fe8-4118-bf32-cf0b4ef29032","apiKey":"22552b96-8783-46a9-b0eb-075ddfa8893e"},"i
rw.infra.svc.cluster.local","port":5432,"database":"iwd","user":"iwd","password":"iwd","ssl":false},"apiKeys":{
A"}}'
```

Deploy Intelligent Workload Distribution

Contents

- 1 Assumptions
- 2 Kubernetes
  - 2.1 Prepare
  - 2.2 Deploy
- 3 Google Kubernetes Engine (GKE)
  - 3.1 Prepare
  - 3.2 Deploy
- 4 Azure Kubernetes Service (AKS)
  - 4.1 Prepare
  - 4.2 Deploy
- 5 Validate the deployment

Learn how to deploy Intelligent Workload Distribution (IWD) 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.

Kubernetes

Prepare

1. Create a new project using the following command:

   ```
   kubectl create namespace iwd
   ```

2. Create a pull secret for accessing the JFrog registry. See Create the pull secret.
3. Download the IWD helm chart from the JFrog repository. See Download the Helm charts.
4. Create a gauth client.
   IWD requires *clientId* and *clientSecret* registered in Authentication Service. These must be provided during Helm Chart deployment. Create new client credentials if they are not already created . Refer to the GWS documentation for more information.

Deploy

1. Extract parameters from chart to see multiple (default) values used to fine tune the installation.

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

   For information on parameters and values in the **values.yaml** file, see Override Helm chart values.
   Sample override file:

   ```
   replicaCount: 1

   image:
     registry: pureengage-docker-staging.jfrog.io
     repository: nexus/iwd
     pullSecrets: []

   gauth:
     auth:
       url: http://gauth-auth.gauth
       redirectUrl: https://gauth.${domain}

   redis:
     nodes: redis://infra-redis-redis-cluster.infra.svc.cluster.local:6379
     useCluster: true
     enableTLS: false
     password:

   gws:
     url: http://gauth-auth.gauth
     clientId:
     clientSecret:
     apiKey:

   ingress:
     enabled: true
     hosts:
       - host: iwd.${domain}
         paths:
           - path: '/iwd/'
             port: 4024
     tls:
       - hosts:
           - iwd.${domain}
         secretName: letsencrypt

   nexus:
     url: http://nexus.nexus
     apikey:

   elasticsearch:
     host: elastic-es-http.infra.svc.cluster.local
     port: 9200
   ```

2. Install IWD using the following command (replace with applicable values):

   ```
   helm install iwd ./iwd-.tgz -f override_values.yaml
   --set gws.clientId=
   --set gws.clientSecret=
   --set redis.password=
   --set nexus.apikey=
   --set gws.apiKey='None'
   --namespace=iwd
   ```

Google Kubernetes Engine (GKE)

Prepare

1. Log in to the GKE cluster.

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

2. Create a new project:
   1. Create a *create-iwd-namespace.json* :

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

   2. Create a namespace using the above JSON:

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

   3. Confirm the namespace creation:

      ```
      kubectl describe namespace iwd
      ```

3. Create a pull secret for accessing the JFrog registry.

   ```
   kubectl create secret docker-registry jfrog-stage-credentials \
    --docker-server=pureengage-docker-staging.jfrog.io \
    --docker-username= \
    --docker-password= \
    --docker-email=
   ```

4. Download the IWD helm chart from the JFrog repository. See Download the Helm charts.
5. Create a gauth client.
   IWD requires *clientId* and *clientSecret* registered in Authentication Service. These must be provided during Helm Chart deployment. Create new client credentials if they are not already created . Refer to the GWS documentation for more information.

Deploy

1. Extract parameters from chart to see multiple (default) values used to fine tune the installation.

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

   For information on parameters and values in the **values.yaml** file, see Override Helm chart values.

Sample override file:

```
replicaCount: 1

image:
  registry: pureengage-docker-staging.jfrog.io
  repository: nexus/iwd
  pullSecrets:
    - name: "pullsecret"

gauth:
  auth:
    url: http://gauth-auth.gws
    redirectUrl: https://gws.nlb02-useast1.gcpe002.gencpe.com

redis:
  nodes: redis://infra-redis-redis-cluster.infra.svc.gke2-useast1.gcpe002.gencpe.com:6379
  useCluster: true
  enableTLS: false
  #password: xxx #in secrets

gws:
  url: http://gauth-auth.gws
  #clientId: xxx #in secrets
  #clientSecret: xxx #in secrets
  #apiKey: xxx #in secrets

ingress:
  enabled: true
  hosts:
    - host: iwd.nlb02-useast1.gcpe002.gencpe.com
      paths:
        - path: '/iwd/'
          port: 4024
  annotations:
    cert-manager.io/issuer-name: ca-cluster-issuer
    kubernetes.io/ingress.class: nginx
  tls:
    - hosts:
      - iwd.nlb02-useast1.gcpe002.gencpe.com
      secretName: iwd-ingress-cert

nexus:
  url: http://nexus.nexus
  #apikey: xxx #in secrets

elasticsearch:
  host: elastic-elasticsearch-master.infra.svc.gke2-useast1.gcpe002.gencpe.com
  port: 9200

monitoring:
  # Deploy ServiceMonitor
  enabled: true
  # Create PrometheusRule k8s object with alarm definitions
  alarms: true
  # Create ConfigMap with Grafana Dashboards
  dashboards: true
```

2. Install IWD using the following command (replace with applicable values):

```
helm install iwd ./iwd-.tgz -f override_values.yaml
--set gws.clientId=
--set gws.clientSecret=
```

```
--set redis.password=
--set nexus.apikey=
--set gws.apiKey='None'
--namespace=iwd
```

Azure Kubernetes Service (AKS)

Prepare

1. Log in to the AKS cluster.

   ```
   az aks get-credentials --resource-group  --name  --admin
   ```

2. Create a new project:
   1. Create a *create-iwd-namespace.json* :

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

   2. Create a namespace using the above JSON:

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

   3. Confirm the namespace creation:

      ```
      kubectl describe namespace iwd
      ```

3. Create a pull secret for accessing the JFrog registry.

   ```
   kubectl create secret docker-registry pullsecret \
   --docker-server=pureengageuse1-docker-multicloud.jfrog.io \
   --docker-username= \
   --docker-password= \
   --docker-email=
   ```

4. Download the IWD helm chart from the JFrog repository. See Download the Helm charts.
5. Create a gauth client.
   IWD requires *clientId* and *clientSecret* registered in Authentication Service. These must be provided during Helm Chart deployment. Create new client credentials if they are not already created . Refer to the GWS documentation for more information.

Deploy

1. Extract parameters from chart to see multiple (default) values used to fine tune the installation.

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

   For information on parameters and values in the **values.yaml** file, see Override Helm chart values.
   Sample override file:

   ```
   replicaCount: 1

   image:
     registry: pureengageuse1-docker-multicloud.jfrog.io
     repository: nexus/iwd
     pullSecrets:
       - name: "pullsecret"

   gauth:
     auth:
       url: http://gauth-auth.${GAUTH_NAMESPACE}
       redirectUrl: https://gauth.${DOMAIN}

   redis:
     nodes: redis://${REDIS_ADDR}:${REDIS_PORT}
     useCluster: true
     enableTLS: false
     #password: xxx #in secrets

   gws:
     url: http://gauth-auth.${GAUTH_NAMESPACE}
     #clientId: xxx #in secrets
     #clientSecret: xxx #in secrets
     #apiKey: xxx #in secrets

   ingress:
     enabled: true
     hosts:
       - host: iwd.${domain}
         paths:
           - path: '/iwd/'
             port: 4024
     annotations:
       cert-manager.io/issuer-name: ca-cluster-issuer
       kubernetes.io/ingress.class: nginx
     tls:
       - hosts:
         - iwd.${domain}
         secretName: iwd-ingress-cert

   nexus:
     url: http://nexus.${NEXUS_NAMESPACE}
     #apikey: xxx #in secrets

   elasticsearch:
     host: ${ES_ADDR}
     port: 9200

   monitoring:
     # Deploy ServiceMonitor
     enabled: true
     # Create PrometheusRule k8s object with alarm definitions
     alarms: true
   ```

```
  # Create ConfigMap with Grafana Dashboards
  dashboards: true
```

2. Install IWD using the following command (replace with applicable values):

```
helm install iwd ./iwd-.tgz -f override_values.yaml
 --set gws.clientId=
 --set gws.clientSecret=
 --set redis.password=
 --set nexus.apikey=
 --set gws.apiKey='None'
 --namespace=iwd
```

Validate the deployment

Watch the helm output at the end of installation. It provides the status and additional information about where to log in to the IWD UI.

See the following sample output:

```
Release "iwd" has been upgraded. Happy Helming!
NAME: iwd
LAST DEPLOYED: Tue Jul 13 10:18:07 2021
NAMESPACE: iwd
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Please be patient while iwd 100.0.0741322 is being deployed
```

Upgrade, roll back, or uninstall

Contents

- 1 Supported upgrade strategies
- 2 Timing
  - 2.1 Scheduling considerations
- 3 Monitoring
- 4 Preparatory steps
- 5 Rolling Update
  - 5.1 Rolling Update: Upgrade
  - 5.2 Rolling Update: Verify the upgrade
  - 5.3 Rolling Update: Rollback
  - 5.4 Rolling Update: Verify the rollback
- 6 Uninstall

Learn how to upgrade, roll back, or uninstall IWD.

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
>
> The instructions on this page assume you have deployed the services in service-specific 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.

Supported upgrade strategies

Intelligent Workload Distribution supports the following upgrade strategies:

Service	Upgrade Strategy	Notes
Intelligent Workload Distribution	Rolling Update	

For a conceptual overview of the upgrade strategies, refer to Upgrade strategies in the Setting up Genesys Multicloud CX Private Edition guide.

Timing

A regular upgrade schedule is necessary to fit within the Genesys policy of supporting N-2 releases, but a particular release might warrant an earlier upgrade (for example, because of a critical security fix).

If the service you are upgrading requires a later version of any third-party services, upgrade the third-party service(s) before you upgrade the private edition service. For the latest supported versions of third-party services, see the Software requirements page in the suite-level guide.

Scheduling considerations

Genesys recommends that you upgrade the services methodically and sequentially: Complete the upgrade for one service and verify that it upgraded successfully before proceeding to upgrade the next service. If necessary, roll back the upgrade and verify successful rollback.

Monitoring

Monitor the upgrade process using standard Kubernetes and Helm metrics, as well as service-specific metrics that can identify failure or successful completion of the upgrade (see Observability in Intelligent Workload Distribution).

Genesys recommends that you create custom alerts for key indicators of failure — for example, an alert that a pod is in pending state for longer than a timeout suitable for your environment. Consider including an alert for the absence of metrics, which is a situation that can occur if the Docker image is not available. Note that Genesys does not provide support for custom alerts that you create in your environment.

Preparatory steps

Ensure that your processes have been set up to enable easy rollback in case an upgrade leads to compatibility or other issues.

Each time you upgrade a service:

1. Review the release note to identify changes.
2. Ensure that the new package is available for you to deploy in your environment.
3. Ensure that your existing **-values.yaml** file is available and update it if required to implement changes.

Rolling Update

Rolling Update: Upgrade

Execute the following command to upgrade :

```
helm upgrade --install  -f -values.yaml  -n
```

Tip: If your review of Helm chart changes (see Preparatory Step 3) identifies that the only update you need to make to your existing **-values.yaml** file is to update the image version, you can pass the image tag as an argument by using the `--set` flag in the command:

```
helm upgrade --install  -f -values.yaml  --set .image.tag=
```

For example,

```
helm upgrade -f ./values.yaml iwd iwd-900.67.1121.tgz -n iwd.
```

Rolling Update: Verify the upgrade

Follow usual Kubernetes best practices to verify that the new service version is deployed. See the information about initial deployment for additional functional validation that the service has upgraded successfully.

Rolling Update: Rollback

Execute the following command to roll back the upgrade to the previous version:

```
helm rollback
```

or, to roll back to an even earlier version:

```
helm rollback
```

Alternatively, you can re-install the previous package:

1. Revert the image version in the `.image.tag` parameter in the **-values.yaml** file. If applicable, also revert any configuration changes you implemented for the new release.
2. Execute the following command to roll back the upgrade:

   ```
   helm upgrade --install  -f -values.yaml
   ```

   Tip: You can also directly pass the image tag as an argument by using the `--set` flag in the command:

   ```
   helm upgrade --install  -f -values.yaml  --set .image.tag=
   ```

Rolling Update: Verify the rollback

Verify the rollback in the same way that you verified the upgrade (see Rolling Update: Verify the upgrade).

Uninstall

> **Warning**
> Uninstalling a service removes all Kubernetes resources associated with that service. Genesys recommends that you contact Genesys Customer Care before uninstalling any private edition services, particularly in a production environment, to ensure that you understand the implications and to prevent unintended consequences arising from, say, unrecognized dependencies or purged data.

Execute the following command to uninstall :

```
helm uninstall  -n
```

For example,

`helm uninstall iwd -n iwd`.

Observability in Intelligent Workload Distribution

Contents

- 1 Monitoring
  - 1.1 Enable monitoring
  - 1.2 Configure metrics
- 2 Alerting
  - 2.1 Configure alerts
- 3 Logging

Learn about the logs, metrics, and alerts you should monitor for Intelligent Workload Distribution.

Related documentation:

-
-
-
-

RSS:

- For private edition

Monitoring

Private edition services expose metrics that can be scraped by Prometheus, to support monitoring operations and alerting.

- As described on Monitoring overview and approach, you can use a tool like Grafana to create dashboards that query the Prometheus metrics to visualize operational status.
- As described on Customizing Alertmanager configuration, you can configure Alertmanager to send notifications to notification providers such as PagerDuty, to notify you when an alert is triggered because a metric has exceeded a defined threshold.

The services expose a number of Genesys-defined and third-party metrics. The metrics that are defined in third-party software used by private edition services are available for you to use as long as the third-party provider still supports them. For descriptions of available Intelligent Workload Distribution metrics, see:

-

See also System metrics.

Enable monitoring

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
	Both or either, depends on harvester	Default is 4024 (overridden by values)	/iwd/v3/metrics	15 sec recommended, depends on harvester

Configure metrics

Metrics are available when requested. No additional configuration is required.

Alerting

Private edition services define a number of alerts based on Prometheus metrics thresholds.

> **Important**
>
> You can use general third-party functionality to create rules to trigger alerts based on metrics values you specify. Genesys does not provide support for custom alerts that you create in your environment.

For descriptions of available Intelligent Workload Distribution alerts, see:

-

Configure alerts

Private edition services define a number of alerts by default (for Intelligent Workload Distribution, see the pages linked to above). No further configuration is required.

The alerts are defined as **PrometheusRule** objects in a **prometheus-rule.yaml** file in the Helm charts. As described above, Intelligent Workload Distribution does not support customizing the alerts or defining additional **PrometheusRule** objects to create alerts based on the service-provided metrics.

Logging

Logging is done to *stdout*.

IWD metrics and alerts

Contents

- 1 Metrics
- 2 Alerts

Find the metrics *No results* exposes and the alerts defined for *No results*.

Related documentation:

-

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval
IWD	Both or either, depends on harvester	Default is 4024 (overridden by values)	/iwd/v3/metrics	15 sec recommended, depends on harvester

See details about:

- IWD metrics
- IWD alerts

Metrics

Metric and description	Metric details	Indicator of
iwd_redis_connections_established Current number of established Redis connections	**Unit:** **Type:** gauge **Label:** **Sample value:** 0	
iwd_redis_connections_reconnecting Current number of reconnecting Redis connections	**Unit:** **Type:** gauge **Label:** **Sample value:** 0	
iwd_redis_connections_ready Current number of ready Redis connections	**Unit:** **Type:** gauge **Label:** **Sample value:** 1	
iwd_redis_duration_until_ready Duration until ready state reached	**Unit:** **Type:** histogram **Label:** 'le' **Sample value:** 0, 1, 39	
iwd_redis_errors_total	**Unit:**	

Metric and description	Metric details	Indicator of
Total number of Redis connection errors	**Type:** counter **Label:** **Sample value:** 0	
iwdTenantDB_db_connect_total The total number of all database connection requests	**Unit:** **Type:** counter **Label:** 'db' **Sample value:** 1252424, 1457770	
iwdTenantDB_db_disconnect_total The total number of all database disconnection requests	**Unit:** **Type:** counter **Label:** 'db' **Sample value:** 1252424, 1457770	
iwdTenantDB_db_request_total The total number of all Database requests sent	**Unit:** **Type:** counter **Label:** 'db' **Sample value:** 4850730, 5056452	
iwdTenantDB_db_success_total The total number of all all Database requests executed successfully	**Unit:** **Type:** counter **Label:** 'db', 'command' **Sample value:** 2307896, 2126805, 1221394, 1450355	
iwdTenantDB_db_errors_total The total number of all Database errors	**Unit:** **Type:** counter **Label:** 'db', 'code' **Sample value:** 131, 5, 4	
iwdTenantDB_db_request_duration_milliseconds Database transaction duration	**Unit:** **Type:** histogram **Label:** 'le', 'db', 'method' **Sample value:** 2290844, 2306385, 2307241, 2307894	
iwd_process_cpu_user_seconds_total Total user CPU time spent in seconds.	**Unit:** **Type:** counter **Label:** **Sample value:** 1634045655571	
iwd_process_cpu_system_seconds_total Total system CPU time spent in seconds.	**Unit:** **Type:** counter **Label:** **Sample value:** 1634045655571	
iwd_process_cpu_seconds_total Total user and system CPU time spent in seconds.	**Unit:** **Type:** counter **Label:** **Sample value:** 1634045655571	
iwd_process_start_time_seconds	**Unit:**	

Metric and description	Metric details	Indicator of
Start time of the process since unix epoch in seconds.	**Type:** gauge **Label:** **Sample value:** 1633992102	
iwd_process_resident_memory_bytes Resident memory size in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_process_virtual_memory_bytes Virtual memory size in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_process_heap_bytes Process heap size in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_process_open_fds Number of open file descriptors.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_process_max_fds Maximum number of open file descriptors.	**Unit:** **Type:** gauge **Label:** **Sample value:** 197176	
iwd_nodejs_eventloop_lag_seconds Lag of event loop in seconds.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_nodejs_active_handles Number of active libuv handles grouped by handle type. Every handle type is C++ class name.	**Unit:** **Type:** gauge **Label:** 'type' **Sample value:** 17, 1, 69	
iwd_nodejs_active_handles_total Total number of active handles.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_nodejs_active_requests Number of active libuv requests grouped by request type. Every request type is C++ class name.	**Unit:** **Type:** gauge **Label:** 'type' **Sample value:** 2	
iwd_nodejs_active_requests_total	**Unit:**	

Metric and description	Metric details	Indicator of
Total number of active requests.	**Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_nodejs_heap_size_total_bytes Process heap size from node.js in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_nodejs_heap_size_used_bytes Process heap size used from node.js in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_nodejs_external_memory_bytes Nodejs external memory size in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
iwd_nodejs_heap_space_size_total_bytes Process heap space size total from node.js in bytes.	**Unit:** **Type:** gauge **Label:** 'space' **Sample value:** 262144, 16777216, 130428928, 6721536	
iwd_nodejs_heap_space_size_used_bytes Process heap space size used from node.js in bytes.	**Unit:** **Type:** gauge **Label:** 'space' **Sample value:** 32808, 1479672, 92634792, 4852384	
iwd_nodejs_heap_space_size_available_bytes Process heap space size available from node.js in bytes.	**Unit:** **Type:** gauge **Label:** 'space' **Sample value:** 0, 6899976, 37040456, 1542496	
iwd_nodejs_version_info Node.js version info.	**Unit:** **Type:** gauge **Label:** 'version', 'major', 'minor', 'patch' **Sample value:** 1	
iwd_request_total The total number of all API requests received	**Unit:** **Type:** counter **Label:** **Sample value:** 177186	
iwd_success_total The total number of all API requests with success response	**Unit:** **Type:** counter **Label:** 'ccid' **Sample value:** 21400, 46769, 48539	

Metric and description	Metric details	Indicator of
iwd_errors_total The total number of all API requests with error response	**Unit:** **Type:** counter **Label:** 'ccid' **Sample value:** 438, 49, 4	
iwd_client_error_total The total number of all API requests with client error response	**Unit:** **Type:** counter **Label:** 'ccid' **Sample value:** 204, 49, 2	
iwd_server_error_total The total number of all API requests with server error response	**Unit:** **Type:** counter **Label:** 'ccid' **Sample value:** 234, 2	
iwd_api_request_total The total number of all API requests	**Unit:** **Type:** counter **Label:** 'method', 'path', 'code', 'ccid' **Sample value:** 3570, 3584, 25079, 19500	
iwd_api_request_long Number of API requests that took long time to execute	**Unit:** **Type:** counter **Label:** **Sample value:**	
iwd_api_request_closed Number of API requests that expired before response was sent	**Unit:** **Type:** counter **Label:** 'method', 'path' **Sample value:** 3, 14, 4, 9	
iwd_api_request_duration_milliseconds API requests duration	**Unit:** **Type:** histogram **Label:** 'le', 'method', 'path', 'code', 'ccid' **Sample value:** 6, 2708, 3502, 3570	
iwd_api_blacklist Total number of blacklisted requests	**Unit:** **Type:** counter **Label:** **Sample value:**	
iwd_cometd_connections_total The current number of client cometd connections to GWS	**Unit:** **Type:** gauge **Label:** 'type', 'ccid' **Sample value:** 27, 1, 41	
iwd_cometd_errors_total The total number of client cometd errors	**Unit:** **Type:** counter **Label:** 'type', 'ccid' **Sample value:** 1	
iwd_cometd_request_errors_tota	**Unit:**	

Metric and description	Metric details	Indicator of
The total number of client cometd error response from GWS	**Type:** counter **Label:** 'type', 'name', 'ccid', 'domain' **Sample value:** 13026, 64, 1, 102	
iwd_cometd_request_current The current number of client cometd requests to GWS	**Unit:** **Type:** gauge **Label:** 'type', 'name', 'ccid', 'domain' **Sample value:** -6318, -11825, 0	
iwd_cometd_request_duration_milliseconds The cometd request duration (to GWS)	**Unit:** **Type:** histogram **Label:** 'le', 'type', 'name', 'ccid', 'domain' **Sample value:** 6298, 6320, 6345, 6395	
iwd_cometd_request_duration_milliseconds_summary The cometd request duration (to GWS): summary	**Unit:** **Type:** summary **Label:** 'quantile', 'type', 'name', 'ccid', 'domain' **Sample value:** 0, 930700, 6577, 89959	
iwd_cometd_events_total The total number of client cometd events from GWS	**Unit:** **Type:** counter **Label:** 'type', 'name', 'ccid', 'domain' **Sample value:** 80, 443, 346, 17	

Alerts

The following alerts are defined for IWD.

Alert	Severity	Description	Based on	Threshold
IWD error rate	CRITICAL	Triggered when the number of errors in IWD exceeds the threshold for 15 min period.		Default number of errors: 2
IWD DB errors	CRITICAL	Triggered when IWD experiences more than 2 errors within 1 minute during operations with database.		Default number of errors: 2
Memory usage is above 3000 Mb	CRITICAL	Triggered when the pod memory usage is above 3000 MB.		Default memory usage: 3000 MB
Database connections above	HIGH	Triggered when pod database		Default number of connections: 75

Alert	Severity	Description	Based on	Threshold
75		connections number is above 75.		