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.

# Digital Channels Private Edition Guide

Deploy

8/25/2025

# Contents

Learn how to deploy 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.

## Deploy AI Connector

> **Important**
> Make sure to review Before you begin for the full list of prerequisites required to deploy Digital Channels AI Connector.

## Prepare your environment

To prepare your environment for the Google Kubernetes Engine (GKE) deployment, complete the steps in this section.

### GKE

Log in to the GKE cluster from the host where you will run the deployment:

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

Create a JSON file called **create-nexus-namespace.json** with the following content:

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

Use the JSON file to create a new namespace for Digital Channels AI Connector:

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

Now, confirm the created namespace:

```
kubectl describe namespace athena
```

### Configure a secret to access JFrog

If you haven't done so already, create a secret for accessing the JFrog registry:

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

## Deploy

To deploy AI Connector, you'll need the Helm package and override files you downloaded in a previous step. Copy **values.yaml** and the Helm package (**athena-.tgz**) to the installation location.

You must override the following key sections in **values.yaml:**

- image.*
- athena.nexus.*
- athena.redis.*
- athena.db.*
- ingress.*

Here's an example of how your **values.yaml** file might look:

```
# Default values for athena.
```

```
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

version: "100.0.124.3419" # AI Connector Version

nameOverride: ""

fullnameOverride: ""

replicaCount: 1

image:
  registry: "pureengage-docker-staging.jfrog.io"
  repository: nexus/athena
  pullPolicy: IfNotPresent
  pullSecrets:
    - name:

serviceAccount:
  create: false
  name: ""
  annotations: {}

podAnnotations: {}

podLabels: {}

podSecurityContext:
  runAsNonRoot: true
  runAsUser: 500
  runAsGroup: 500
  fsGroup: 500

securityContext: {}

configChecksum: true

secretChecksum: true

containerPort: 4084

service:
  enabled: true
  type: ClusterIP
  annotations: {}
  port: 80

ingress:
  enabled: false
  annotations: {}
  hosts:
    - host: athena.local
      paths: []
  tls: []
  #  - secretName: athena-tls-secret
  #    hosts:
  #      - athena.local

resources: {}
  # limits:
  #   cpu: 100m
  #   memory: 128Mi
  # requests:
```

```
  #   cpu: 100m
  #   memory: 128Mi

nodeSelector: {}

tolerations: []

affinity: {}

priorityClassName: ""

dnsPolicy: ClusterFirst

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

monitoring:
  enabled: false

athena:
  server:
    apiPrefix: "/nexus/v3"
  nexus:
    url: ""
    apiPrefix: "/nexus/v3"
    apiKey: ""
    timeout: 10000
  db:
    host: ""
    port: 5432
    user: ""
    password: ""
    database: ""
    ssl: false
  redis:
    nodes: "redis://:6379"
    password: ""
    cluster: true
    tls: false
  google:
    speechApiKey: ""
```

### Run the following command to install AI Connector:

```
helm upgrade --install  /athena-.tgz -f values.yaml
```

## Validate the deployment

To validate the deployment, first run the following code snippet

```
kubectl port-forward service/athena :80
```

Then, send the GET request on the following URL:

```
$athenaURL/health/detail
```

where **$athenaURL** is the fully qualified domain name (FQDN) for AI Connector.

### The response should look like this:


```
{
    "buildInfo": {
        "version": "100.0.001.97446",
        "changeset": "565f432fa8f4555276b55e8237cebcfb201b986e",
        "timestamp": "Mon Jan 17 10:21:22 UTC 2022"
    },
    "startTime": "2022-03-17T13:15:22.873Z",
    "upTime": 49338032,
    "os": {
        "hostname": "athena-6bb9c5c68f-bz449",
        "upTime": 52366.39,
        "freemem": 1316397056,
        "loadavg": [0.35, 0.37, 0.63],
        "totalmem": 4124729344
    },
    "memoryUsage": {
        "rss": 178757632,
        "heapTotal": 83382272,
        "heapUsed": 80987072,
        "external": 1890524,
        "arrayBuffers": 126610
    },
    "redis": {
        "state": "READY",
        "latency": 5
    },
    "db": {
        "latency": 202
    },
    "isReady": true
}
```


### The deployment is successful if `state="green"`. You can also confirm that `db.ready=true` and `redis.ready=true`.

## Uninstall

Execute the following command to uninstall AI Connector:

```
helm delete  -n
```

## Next steps

Complete the steps in Provisioning overview to finish deploying AI Connector.