Oracle® Communications Billing and Revenue Management

AI Deployment and Integration Guide



Release 16.26.0
G54437-01
October 2026



Oracle Communications Billing and Revenue Management AI Deployment and Integration Guide, Release 16.26.0

G54437-01

Copyright © 2026, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Contents

About This Content

1 Getting Started with AI in Cloud Scale Monetization

Overview of the Deployment Package 1
Tasks for Preparing Your Cloud Native Environment 1
Downloading Packages for the Cloud Native Helm Charts and Docker Files 2
Pulling Images from the Oracle Container Registry 2
Downloading Images from Oracle Software Delivery Website 3
Monitoring AI Services 3
Configuring Grafana Dashboards for AI Services 4

2 About Prerequisite Software Requirements

System Requirements 1
Setting Up Prerequisite Software and Tools 1

3 Installing AI Services for Cloud Scale Monetization

Overview of Installation Tasks 1
Creating Object Storage Buckets in OCI 2
Creating Data Flow Private Endpoints 2
Generating API Keys 3
Creating the archive.zip File 3
Uploading Files to Object Storage 4
Creating a Data Flow Application 4
Setting Up a Data Science Job 5

4 Deploying and Configuring the Data Service

Deploying the Data Service 1
Configuring and Using the Data Service 4
Configuring Multiple Data Sources 4

Copyright © 2026, Oracle and/or its affiliates.

Collecting Metrics for Anomaly Detection 5

5 Deploying and Configuring the Training Service

Deploying the Training Service 1
Configuring and Using the Training Service 5
Training a Multi-Head Offer Recommendation Model 5
Training an Anomaly Detection Model 6

6 Deploying and Configuring the Prediction Service

Deploying the Prediction Service 1
Configuring and Using the Prediction Service 4
Configuring the MCP Interface 4
Implementing Siebel Next Best Offer Recommendations 5
Predicting Time-Series Anomalies 6

7 Deploying and Configuring the Billing Care AI Agent

Deploying the Billing Care AI Agent 1
Deploying the Billing Care AI Agent in Kubernetes 1
Deploying the Billing Care AI Agent by Using OCI Generative AI Hosted Deployment 5
Using the Billing Care AI Agent 7

Copyright © 2026, Oracle and/or its affiliates.

About This Content

Describes how to integrate AI-based microservices with Oracle Monetization Suite.

Audience

This guide is intended for developers.

Copyright © 2026, Oracle and/or its affiliates.

1

Getting Started with AI in Cloud Scale Monetization

Learn how to get started with artificial intelligence (AI) and Machine Learning (ML) using the deployment assets provided for Oracle Communications Cloud Scale Monetization. These assets include container images, Helm charts, configuration files, and other components that assist you in installing, deploying, and managing AI services on Kubernetes.

You can use these resources in both Oracle Cloud Infrastructure (OCI) and other supported environments.

Topics in the document:

- Overview of the Deployment Package
- Tasks for Preparing Your Cloud Native Environment
- Downloading Packages for the Cloud Native Helm Charts and Docker Files
- Pulling Images from the Oracle Container Registry
- Downloading Images from Oracle Software Delivery Website
- Monitoring AI Services
- Configuring Grafana Dashboards for AI Services

Overview of the Deployment Package

The deployment package provides all components required to deploy, manage, and monitor data, training, and prediction services for AI-based workflows on Kubernetes. The package includes:

- Ready-to-use Docker images and Helm charts for each service (data service, training service, and prediction service), allowing you to orchestrate containers in your Kubernetes environment. Update the **values.yaml** file as needed to reflect your specific configuration requirements.
- Dockerfiles and build files that enable you to apply minor updates to the base operating system or add additional dependencies.
- Sample configuration files, Python scripts, and shell scripts that serve as references for customizing your deployment framework.
- Sample Grafana dashboard JSON files for each service, designed to streamline performance monitoring and observability with Prometheus and Tempo.

Tasks for Preparing Your Cloud Native Environment

Prepare your system for the BRM cloud native deployment by performing the following high-level tasks:

1. Downloading the Helm charts for the cloud native deployment. See "Downloading Packages for the Cloud Native Helm Charts and Docker Files".

Copyright © 2026, Oracle and/or its affiliates.

2. Downloading the cloud native images in one of these ways:
   a. From the Oracle Container Registry. To do so, see "Pulling Images from the Oracle Container Registry".
   b. From the Oracle Software Delivery website. To do so, see "Downloading Images from Oracle Software Delivery Website".

Downloading Packages for the Cloud Native Helm Charts and Docker Files

To download the packages for the cloud native helm charts and docker files, see "Downloading Packages for the BRM Cloud Native Helm Charts" in *BRM Cloud Native Deployment Guide*.

Table 1-1 lists the package name and archive files that need to be extracted for AI and ML services.

Table 1-1 Packages and Archive Files

Package Name	Archive Files
Oracle Communications Cloud Native ML Helm Charts 16.26.0.0.0	**ml-helmcharts-16.26.0.0.0.tgz** Includes: • **data-fetch-16.26.0.tgz** • **train-utility-16.26.0.tgz** • **prediction-service-16.26.0.tgz**
Oracle Communications Cloud Native ML Docker files and Artifacts Bundle 16.26.0.0.0	**ml-dockerfiles-artifacts-16.26.0.0.0.tgz** Includes: • **data_orchestrator_docker_bundle.tar.gz** • **data_processor_docker_bundle.tar.gz** • **train_orchestrator_docker_bundle.tar.gz** • **train_processor_docker_bundle.tar.gz** • **predict_orchestrator_docker_bundle.tar.gz** • **predict_service_docker_bundle.tar.gz**
Oracle Communications Cloud Native BRM Billing Agent Helm Charts 16.26.0.0.0	**brm-billing-agent-helmcharts-16.26.0.0.0.tgz** Includes: • **brm-billing-agent-16.26.0.tar.gz**
Oracle Communications Cloud Native BRM Billing Agent Docker files and Artifacts Bundle 16.26.0.0.0	**brm-billing-agent-dockerfiles-artifacts-16.26.0.0.0.tgz** Includes: • **brm-billing-agent-dockerfiles-artifacts.tgz**

Pulling Images from the Oracle Container Registry

To pull images from the Oracle Container Registry, see "Pulling BRM Images from the Oracle Container Registry" in *BRM Cloud Native Deployment Guide*.

Table 1-2 lists the image names for the AI-ML cloud native components.

Copyright © 2026, Oracle and/or its affiliates.

Table 1-2 Cloud Native Images

Component Name	Image Name
ML Data Fetch Orchestrator	**ml-data-fetch-orchestrator**
ML Data Fetch Processor	**ml-data-fetch-processor**
ML Train Utility Orchestrator	**ml-train-utility-orchestrator**
ML Train Utility Processor	**ml-train-utility-processor**
ML Prediction Orchestrator	**ml-prediction-orchestrator**
ML Prediction Service	**ml-prediction-service**
BRM Billing Agent	**brm-billing-agent**

Downloading Images from Oracle Software Delivery Website

To download images from Oracle Software Delivery Website, see "Downloading BRM Images from the Oracle Software Delivery Website" in *BRM Cloud Native Deployment Guide*.

Table 1-3 lists the names of the packages and package files that you need to extract for using and configuring AI and ML services.

Table 1-3 Cloud Native Package and Package Files

Package Name	Package File Name
Oracle Communications Cloud Native ML Data Service Images 16.26.0.0.0	**ml-data-fetch-16.26.0.0.0.tgz** Includes: • **ml-data-fetch-orchestrator.tar** • **ml-data-fetch-processor.tar**
Oracle Communications Cloud Native ML Training Service Images 16.26.0.0.0	**ml-train-utility-16.26.0.0.0.tgz** Includes: • **ml-train-utility-orchestrator.tar** • **ml-train-utility-processor.tar**
Oracle Communications Cloud Native ML Prediction Service Images 16.26.0.0.0	**ml-prediction-16.26.0.0.0.tgz** Includes: • **ml-prediction-orchestrator.tar** • **ml-prediction-service.tar**
Oracle Communications Cloud Native BRM Billing Agent Image 16.26.0.0.0	**brm-billing-agent-16.26.0.0.0.tgz** Includes: • **brm-billing-agent.tar**

Monitoring AI Services

You can monitor all the AI services using service metrics. These metrics allow you to track and monitor the total number of requests and the time taken by the services.

Table 1-4 lists information about the available metrics for the services.

Copyright © 2026, Oracle and/or its affiliates.

Table 1-4 Service Metrics

Metric Name	Description
aiml_data_service_request_seconds	Total time taken by the latest data service request.
aiml_data_service_request_seconds_count	Total number of data service requests received.
aiml_data_service_request_seconds_max	Maximum time taken by a data service request to process.
aiml_data_service_request_seconds_sum	Total time taken by all the data service requests to finish processing.
aiml_train_service_request_seconds	Total time taken by the latest training service request.
aiml_train_service_request_seconds_count	Total number of training service requests received.
aiml_train_service_request_seconds_max	Maximum time taken by a training service request to process.
aiml_train_service_request_seconds_sum	Total time taken by all the training service requests to finish processing.
predict_request_latency_seconds	Total time taken by the latest prediction service request.
predict_request_seconds_count	Total number of prediction service requests received.
predict_request_seconds_max	Maximum time taken by a prediction service request to process.
predict_request_seconds_sum	Total time taken by all the prediction service requests to finish processing.

In addition to these, all the metrics that are provided by Micrometer, Tempo, and Kubernetes are available.

Configuring Grafana Dashboards for AI Services

You can create dashboards in Grafana to display the metric data for your AI services.

Alternatively, you can use the sample dashboards included in the **ml-helmcharts-16.26.0.0.0.tgz** package.

Table 1-5 lists the sample dashboards.

Table 1-5 Sample Grafana Dashboards

File Name	Description
data-fetch-grafana-dashboard.json	Allows you to view metrics for the data service requests.
train-utility-grafana-dashboard.json	Allows you to view metrics for the training service requests.
prediction-grafana-dashboard.json	Allows you to view metrics for the prediction or recommendation service requests.

Copyright © 2026, Oracle and/or its affiliates.

2

About Prerequisite Software Requirements

Learn about the required operating system, software, tools, and files required to deploy artificial intelligence (AI) and machine learning (ML) services in Oracle Communications Cloud Scale Monetization.

Topics in this document:

- System Requirements
- Setting Up Prerequisite Software and Tools

System Requirements

AI and ML services in Oracle Monetization Suite require a Linux operating system. To ensure efficient deployment and operation, use the current, supported versions of all required software, tools, platforms, and reference files.

For details and a list of supported versions, see *BRM Compatibility Matrix*.

Setting Up Prerequisite Software and Tools

Before you deploy data, training, and prediction services, you must install and configure the following software and tools:

1. **Kubernetes Cluster**: Set up a Kubernetes cluster to orchestrate and manage your containerized service deployments.

   For more information about Kubernetes, see "Kubernetes Concepts" in the Kubernetes documentation.
2. **Container Runtime**: Install a Kubernetes-supported container runtime such as Podman on all cluster nodes to support containerized workloads.
3. **Helm 3**: Install Helm 3 to deploy and manage services using Helm charts.

   For installation instructions, see "Installing Helm" in the Helm documentation.
4. **Kubernetes Namespaces**: Create the following namespaces for separation of services:
   - **data-fetch-service** for the data service
   - **training-utility-service** for the training service
   - **predict-service** for the prediction service
5. **Oracle Cloud Infrastructure (OCI) Policies**: If you plan to deploy services in OCI, create and assign necessary OCI policies to enable access between Object Storage, Logging, Data Flow, and Data Science.

   A policy defines who has access to specific OCI resources and what actions they can perform. For more information, see "Getting Started with Policies" in *Oracle Cloud Infrastructure Documentation*.

Copyright © 2026, Oracle and/or its affiliates.

> **Note**
>
> This step is required only for deployments in OCI.

6. **Oracle Identity Cloud Service (IDCS)**: Obtain access to IDCS, which serves as an identity provider for user authentication and access management.

   For more information, see "Federating with Identity Providers" in the *OCI documentation*.
7. **Prometheus**: Install Prometheus to monitor service metrics and trigger alerts.
8. **Grafana**: Install Grafana for visualizing metrics, importing dashboards, and analyzing data.
9. **Tempo**: Install Tempo for distributed tracing to monitor and troubleshoot services.
10. (Optional) **NGINX Ingress Controller**: Install the NGINX Ingress Controller.

    For installation instructions, see Installation Guide in the Kubernetes documentation.

> **Note**
>
> If you are not using the NGINX Ingress Controller, configure direct HTTPS with an SSL certificate and private key. This is the recommended approach.

Copyright © 2026, Oracle and/or its affiliates.

3

Installing AI Services for Cloud Scale Monetization

Learn how to deploy data, training, and prediction services for artificial intelligence (AI) and machine learning (ML) in Oracle Cloud Infrastructure (OCI).

Topics in this document:

- Overview of Installation Tasks
- Creating Object Storage Buckets in OCI
- Creating Data Flow Private Endpoints
- Generating API Keys
- Creating the archive.zip File
- Uploading Files to Object Storage
- Creating a Data Flow Application
- Setting Up a Data Science Job

Overview of Installation Tasks

To install and deploy data service, training service, and prediction service, you need to perform some common installation tasks. These installation tasks are required only if you plan to deploy these services in the OCI environment. These tasks help you to set up the environment.

The high-level installation tasks are:

1. Create a bucket in OCI Object Storage to store configuration files and resources.

   See "Creating Object Storage Buckets in OCI".
2. Create a private endpoint in OCI Data Flow.

   See "Creating Data Flow Private Endpoints".
3. Generate and download public and private API keys.

   See "Generating API Keys".
4. Create and verify the **archive.zip** deployment file.

   See "Creating the archive.zip File".
5. Create any required Python scripts to run in OCI Data Flow.

   > **Note**
   >
   > Complete this step only if you plan to use OCI Data Science for job creation.

6. Upload configuration and dependency files to OCI Object Storage.

   See "Uploading Files to Object Storage".

Copyright © 2026, Oracle and/or its affiliates.

7. Create a Data Flow application in OCI.

   See "Creating a Data Flow Application".

8. Create jobs in OCI Data Science as needed.

   See "Setting Up a Data Science Job".

> **Note**
>
> Complete this step only if you plan to use OCI Data Science for job creation.

9. Set up an Oracle Identity Cloud Service (IDCS) instance to manage authentication.

   See "Federating with Identity Providers" in the *OCI documentation*.

> **Note**
>
> Using IDCS is optional but highly recommended to handle authorization and authentication. You can configure IDCS, and all provided services are certified for use with it.

Creating Object Storage Buckets in OCI

Create buckets in OCI Object Storage to store various configuration files, API keys, scripts, and other resources.

To create a bucket in OCI Object Storage:

1. From the Navigation menu, select **Storage**.

   The Storage pane appears on the right.

2. Click **Buckets**.

   The Buckets page appears.

3. Click **Create Bucket**.

   The Create Bucket page appears.

4. Enter a name for your bucket and use the default values for other settings.

5. Click **Create Bucket**.

Creating Data Flow Private Endpoints

Create a private endpoint in OCI Data Flow:

1. From the Navigation menu, select **Analytics & AI**.

   The Analytics & AI pane appears on the right.

2. Select **Data Flow**.

   The Data Flow page appears.

3. From the left navigation pane, select **Private Endpoints**.

   The Private Endpoints page appears.

Copyright © 2026, Oracle and/or its affiliates.

4. Click **Create private endpoint**.

   The Create private endpoint page appears.

5. Enter a name in the **Name** field.
6. From the list, select the VCN and Subnet if you plan to use a private network. For a public network, you can leave this field empty.
7. In the **DNS zones to resolve** field, add the DNS whose servers need access.

   Fill in additional fields as needed based on your requirements.

8. Click **Create**.

Generating API Keys

To generate and download API keys:

1. Click your profile in the OCI Console and select **User settings**.

   The User settings page appears.

2. In the left panel, select **API Keys**.

   The API Keys page appears.

3. Click **Add API Keys**.
4. Download both the **Private Key** and **Public Key** files.

> **Note**
>
> The **Private Key** file is used for all future authorization of OCI features.

Creating the archive.zip File

To prepare the **archive.zip** deployment package:

1. Pull the dependency packer Docker image:

   ```
   docker pull phx.ocir.io/axmemlgtri2a/dataflow/dependency-packager-linux_arm64_v8:latest
   ```

> **Note**
>
> You can also use Podman, if you prefer.

2. Create a **requirements.txt** file listing the required Python dependencies. For this use case, include **oci** and **pandas**.
3. Place the required JAR files and configuration files in the same directory as **requirements.txt**.
4. Run the following command (using your Python version) to create **archive.zip**:

   ```
   docker run --platform linux/amd64 --rm -v $(pwd):/opt/dataflow --pull always -it phx.ocir.io/axmemlgtri2a/dataflow/dependency-packager-linux_x86_64:latest -p 3.11
   ```

5. Verify that **archive.zip** includes **ojdbc17.jar** and the configuration file.

Copyright © 2026, Oracle and/or its affiliates.

6. Update the configuration file with your User OCID, Tenancy, Region, and other required values.

Uploading Files to Object Storage

To upload files to your Object Storage bucket:

1. In the OCI Console, search for and navigate to **Buckets** and select **Buckets** from the Services results.

   The Buckets page appears with a list of available buckets.
2. Select your bucket.
3. Click **Upload objects**.
4. In the upload wizard:
   a. In the **Select files** section:
      - (Optional) In **Object name prefix**, enter a prefix to prepend it to each uploaded file name.

        To organize files in folders or subfolders, start the prefix with a slash (“/”). For example, `/myfiles` stores files in a `myfiles` folder.
      - Select **Standard** for the storage tier field.
      - Drag your file to the upload area or click to select it.
      - Click **Next**.
   b. In the **Review files** section:
      - Confirm the details and files to upload.
      - Click **Next**.
   c. In the **Upload files** section:
      - Click **Upload objects**.
      - Wait for the upload to complete.
      - Click **Close** when the process finishes.
      - Verify that your file appears in the Objects list for the bucket.

Creating a Data Flow Application

To create an application in OCI Data Flow, perform the following steps:

1. Go to the OCI Console.
2. From the Navigation menu, select **Analytics & AI**.

   The Analytics & AI window appears on the right pane.
3. In the Analytics & AI window, select **Data Flow**.

   The Data Flow page appears.
4. Click **Create Application**.

   The create application details page appears.
5. From the **Spark version** list, select the required version.

Copyright © 2026, Oracle and/or its affiliates.

> **Note**
>
> Oracle recommends using the latest version of Spark for best results.

6. In the **Driver Shape** and **Executer Shape** fields, select the required shape or hardware template. It should have a minimum of 2 OCPU and 32 GB of memory.
7. In the **Number of Executors** field, select **1** or more.
8. In the **Language** field, select **Python**.
9. In **Select a file**, select your bucket and the **dataflow.py** file.

> **Note**
>
> You can locate the **dataflow.py** file in the artifacts provided with the package and you can configure this file as per your requirements.

10. In the **Archive URI** section, select your bucket and the **archive.zip** file.

    See "Creating the archive.zip File" for more information.
11. In **Application log location**, select your bucket for log storage.
12. Click **Create**.

    The data flow application is created.

Setting Up a Data Science Job

> **Note**
>
> The steps for creating training jobs may vary. For details, see "Deploying the Training Service".

To create and set up a job in OCI Data Science, perform the following tasks:

1. Go to the OCI Console.
2. From the Navigation menu, select **Analytics & AI**.

   The Analytics & AI window appears on the right pane.
3. Select **Data Science**.

   The Data Science page appears.
4. Click **Create project**.

   The Create project page appears.
5. Enter the required **Name** and **Description** for your project, then click **Create**.
6. Upload the **oci_config** file, **private_key** file, and **data_flow_config** files to your object storage bucket.

Copyright © 2026, Oracle and/or its affiliates.

> **Note**
>
> You can locate these files in the artifacts provided with the package and you can configure them as per your requirements.

For more information, see "Uploading Files to Object Storage".

7. Go to your project, select **Jobs**, and click **Create job**.
8. In the **Basic Information** section:
   a. Select **Single Node**.
   b. Choose the appropriate compartment.
   c. Enter the required **Name** and **Description** for your job.
9. In the **Configuration** section, set the following environment variables:
   a. **JOB_RUN_ENTRYPOINT**: **controller.py**
   b. **CONDA_ENV_SLUG**: **python_p312_any_x86_64_v1** (for the data service) or **tensorflow216_p310_gpu_v1** (for Training Service)
   c. **CONDA_ENV_TYPE**: **service**
10. In the **Compute Shape** section, select the required shape.

> **Note**
>
> You must choose a compute shape with at least 1 core CPU and 16 GB RAM.

11. In the **Storage** section, insert the required storage.

> **Note**
>
> You need to have at least 50 GB storage.

12. In the **Networking** section, select the appropriate VCN and subnet configurations if you plan to use a private network. For public network, you can leave this field empty.
13. In the **Upload job artifact** section, select the **collect_data_from_dataflow.zip** file.

> **Note**
>
> You can locate the **collect_data_from_dataflow.zip** file in the artifacts provided with the package.

14. In the **Additional Configuration** section, specify the log group and logs as required.
15. In the **Storage Mount** section, click **+ Add object storage mount**.
   a. Select the correct **Compartment** and **Bucket** from the list.
   b. Leave the **Object name prefix** blank.
   c. In the **Destination path and directory** field, enter **/home**.

Copyright © 2026, Oracle and/or its affiliates.

16. Click **Create**.

    The data science job is created.

Copyright © 2026, Oracle and/or its affiliates.

4

Deploying and Configuring the Data Service

Learn how to deploy and configure the data service for Oracle Monetization Suite in both Oracle Cloud Infrastructure (OCI) and non-OCI environments.

Topics in this document:

- Deploying the Data Service
- Configuring and Using the Data Service
- Configuring Multiple Data Sources
- Collecting Metrics for Anomaly Detection

Deploying the Data Service

You can deploy the data service in an OCI environment or in a non-OCI environment.

- Before deploying and configuring the data service, ensure that you have all the prerequisites in place.
  - For OCI deployments, you do this by performing the instructions in "Overview of Installation Tasks".
  - For non-OCI deployments, set up your environment according to your product instructions.
- Then, for both OCI and non-OCI environments, deploy the data service using Helm charts as described below.

To deploy the data service using Helm charts:

1. Download the data service Helm charts from the provided deployment package.

   For more information, see "Downloading Packages for the Cloud Native Helm Charts and Docker Files" and "Setting Up Prerequisite Software and Tools".
2. Create a new key and generate an SSL certificate to run the data service over HTTPS.
3. (Optional) Install the NGINX Ingress Controller:
   - Create a Kubernetes TLS secret using the certificate you generated and your private key:

     ```
     kubectl create secret tls secretName --cert=certificateName.crt --key=privateKeyName.key
     ```
   - Add the ingress-nginx Helm repository:

     ```
     helm repo add ingress-nginx
     ```
   - Update the Helm repository:

     ```
     helm repo update
     ```
   - Install the Ingress Controller:

     ```
     helm install ingress-nginx ingress-nginx/ingress-nginx --namespace nginxNamespace --create-namespace
     ```


Copyright © 2026, Oracle and/or its affiliates.

- Use the following command to attach your service to the NGINX controller:

```
helm install ingress-nginx ingress-nginx/ingress-nginx --namespace
nginxNamespace \
   --set controller.service.enableHttp=false \
   --set controller.service.enableHttps=true \
   --set controller.service.ports.https=443 \
   --set controller.service.nodePorts.https=31231 nodePort \
   --set controller.config.ssl-redirect=true \
   --set controller.config.force-ssl-redirect=true \
   --set controller.ingressClassResource.name=ingressClassName \
   --set controller.ingressClass=ingressClassName \
   --create-namespace
```

4. Update the **values.yaml** file for your deployment:
   a. Set mount paths:
      - **mountPath.dataFilesHostPath**: Specify the path where data files are stored. This field is required and is identified as the default log-file location. For a single worker node, specify a path on that node, such as **/mnt/data_service**. For multiple worker nodes, specify the NFS or OCI File Storage export path. When **tech_choice** is **spark** and **storage_type** is **pvc**, data files are stored under this directory.
      - **mountPath.logFilesHostPath**: Specify a different path if log files are to be stored separately. For multiple worker nodes, specify a path in the shared export.
      - Set folder permissions:

```
groupadd -g 10001 oracle
useradd -mr -u 10001 -g oracle oracle
chown 10001:10001 -R mountPath
```

   b. Set storage configuration:
      - **modelFilesStorage.storageType**: Set to **hostPath** for files on a single worker node, **nfs** for an NFS export shared across worker nodes, or **ociFss** for an OCI File Storage export shared across worker nodes. The default is **nfs**.
      - **modelFilesStorage.nfs.server**: For NFS storage, specify the DNS name or IP address of the NFS server. Ensure that the export is reachable from the worker nodes.
      - **modelFilesStorage.nfs.existingClaim**: When upgrading a deployment that uses an existing PVC, specify its current claim name.
      - **logFileStorage.existingClaim**: When upgrading a deployment that uses a separate existing log-file PVC, specify its current claim name.
      - **modelFilesStorage.ociFss.fileSystemOcid**: For OCI File Storage, specify the file system OCID.
      - **modelFilesStorage.ociFss.mountTargetIp**: For OCI File Storage, specify the mount target IP address.
   c. If you are not using NGINX Ingress, configure HTTPS:
      - **https.enabled**: Set to **true**.
      - **https.source**: Set to **hostPath** when Kubernetes secrets are used for certificate ingestion, or to **chartFiles** to read the certificate and key from chart files.
      - **https.orchestratorServicePort**: Specify the port on which the service is exposed.

Copyright © 2026, Oracle and/or its affiliates.

- **https.path**: If **https.source** is **hostPath**, specify the host path containing the certificate and key.
- **https.certFile** and **https.keyFile**: If **https.source** is **hostPath**, specify the certificate and key file names.
- **https.certPath** and **https.keyPath**: If **https.source** is **chartFiles**, specify the certificate and key files in the **helm/data-fetch/files/certs/** directory.

d. Add **imageRepository** and image tags for **dataFetchOrchestrator** and **dataFetchProcessor**.

e. Under the **dataFetchProcessor.configurableFile** entry, specify the following files:

- **filterFile**: Specify the path to the Python script used to filter data for a **/fetch** request. Place it in **helm/data-fetch/files/scripts/** and specify its file name. The script can import additional files as required.
- **dataSourceFile**: Specify the path to the configuration file containing data-source and OCI connection information. Place it in **helm/data-fetch/files/config/**.
- **logConfigFile**: Specify the path to the configuration file used to change the log level or format. Place it in **helm/data-fetch/files/config/**

Specify these paths relative to the chart's mount path. If a configurable file is larger than 1 MB, place it under **mountPath.dataFilesHostPath** and specify its absolute path inside the container.

> **Note**
>
> Configurable sample files are available for each of the files mentioned above in the artifacts provided with the package: **sample_filter_data.py** (for filterFile), **dnn_preprocess.py** (for additional files), **config.json** (for dataSourceFile), **oci_config** (for ociConfigFile), and **logging_config.py** (for logConfigFile).

f. If your deployment uses OCI services, configure OCI access:

- **ociConfig.enabled**: Set to **true** if the data service uses OCI Data Flow, Data Science, or Object Storage.
- **ociConfig.configPath** and **ociConfig.privateKeyPath**: Specify the relative paths of the OCI configuration file and private key, such as paths under **files/secrets/oci/**.

g. If you are using NGINX, set the NGINX Ingress values:

- **nginx.enabled**: Set to **true**.
- **host**: Set to the host name where the service is deployed.
- **tlsSecretName**: Set to the name of the Kubernetes TLS secret created in step 3.
- **ingressClassName**: Set to your ingress class name.

h. Set **identityURI**, **issuer**, and **audience** to your IDCS configuration. Do one of the following:

- To use IDCS, set **dataFetchOrchestrator.idcs.enabled** to **true**.
- To disable IDCS, set **dataFetchOrchestrator.idcs.enabled** to **false**.

Copyright © 2026, Oracle and/or its affiliates.

5. Install the Data Service Helm charts:

```
helm upgrade --install data-fetch-services helm/data-fetch/ --
namespace=data-fetch-service
```

You can schedule the data fetching process at regular intervals. To do this, configure a cron job to run a suitable shell script.

A sample script, **sample_bash.sh**, is provided with the package. This script sets **delta_data** to **false** for the first run and to **true** for all subsequent runs, so that after the first fetch, only changed data is fetched.

Before using the script, set it to be executable.

Configuring and Using the Data Service

You interact with the data service using REST APIs and can connect other systems through API calls.

For the data service, use the following endpoints to provide data to downstream machine learning and analytical tasks:

- **Collect Data from Source** (**/data/collect**)

  Collects data from specified sources, such as Billing and Revenue Management databases or your custom databases.

- **Fetch Data from Cache** (**/data/fetch**)

  Retrieves collected data based on request parameters and delivers it to required clients.

- **Cache Persisted Data** (**/data/cache**)

  Caches fetched data for fast, repeated access.

These APIs are designed to collect, cache, and make data available for downstream machine learning and analytical tasks.

For more information, see *REST API Reference for AI Services*.

Configuring Multiple Data Sources

You can configure the data service to collect data from multiple sources in one request. For example, you can connect to BRM, Siebel, and Prometheus as data sources and use the collected data together to generate a next best offer recommendation.

To configure multiple data sources:

1. Identify the required data and respective data sources to configure.

   You can configure BRM to provide account, usage, product, service, launch, and related billing data. Similarly, you can configure Siebel to provide customer, purchase, campaign, offer, and label data.

2. Configure BRM and Siebel as separate data sources in the data service configuration file.
3. Specify the source type, query configuration, and connection details:
   - In the input JSON request, add the data sources to the `data_source_type` list.
   - In the input JSON request, add the query configuration for each data source to the `query_builder` list.

Copyright © 2026, Oracle and/or its affiliates.

- In the configuration file, add the connection details for each data source to the `data_source_details` list.

Ensure that the entries in the `data_source_type`, `query_builder`, and `data_source_details` lists use the same order. For example, the connection details in the first `data_source_details` entry apply to the first source type and first query-builder entry.

4. Use the collect API to collect and prepare the data.

   The data service processes the data sources and persists refined data fragments for downstream processing.

5. Use the cache API to cache the data required by the prediction service.

   The training service uses the persisted data fragments to train models.

6. Use the Fetch API to retrieve the data required for a customer.

   The prediction service uses the retrieved data to generate predictions.

> **Note**
>
> If a data source is unavailable, the data service returns an error. Verify the source connection details, credentials, query configuration, and source order before retrying the request.

Collecting Metrics for Anomaly Detection

You can configure the data service to collect historical metrics from Prometheus for training a time-series anomaly detection model. The data service persists the collected metric data for use by the training service.

To collect metrics from Prometheus:

1. Use Prometheus to identify the available metric names, and select the metrics to collect in the query configuration.
2. Configure Prometheus as a data source in the data service configuration file.
3. In the query_builder entry for the Prometheus data source in the Data Service configuration file, specify the query in the following format:

   ```
   filters|start|end|step|metrics
   ```

   where:

   - `filters` is optional. Specify filters as comma-separated key-value pairs: `key1=value1,key2=value2`.
   - `start` and `end` are mandatory. Specify the collection start and end times in ISO 8601 format.
   - `step` is mandatory. It specifies the interval between collected data points.
   - `metrics` is optional. Specify metric names as a comma-separated list: `metric_name1,metric_name2`.

   Use filters to limit collection to a particular resource, such as a pod or virtual machine. If you specify only `start|end|step`, the query collects all available metrics from Prometheus without filtering.

4. Use the collect API to collect the metrics.

Copyright © 2026, Oracle and/or its affiliates.

The data service retrieves the metrics from Prometheus and persists the collected data for downstream processing.

5. Use the persisted metric data to train the anomaly detection model.

   If labels can be generated automatically, you can use the collected data directly for training. Otherwise, manually label the collected data before training. The Training Service uses the metric data and labels to train the model.

Copyright © 2026, Oracle and/or its affiliates.

5

Deploying and Configuring the Training Service

Learn how to deploy and configure the training service for artificial intelligence (AI) and machine learning (ML) in Oracle Monetization Suite within Oracle Cloud Infrastructure (OCI) and non-OCI environments.

Topics in this document:

- Deploying the Training Service
- Configuring and Using the Training Service
- Training a Multi-Head Offer Recommendation Model
- Training an Anomaly Detection Model

Deploying the Training Service

You can deploy the training service either on Oracle Cloud Infrastructure or on a non-OCI environment according to your requirements.

For OCI deployments:

- Complete all common installation tasks described in "Overview of Installation Tasks".
- When uploading files to Object Storage, also upload the following:
  - **OCI configuration file**
  - **Private key file**
  - **Preprocessing script** (Optional)

> **Note**
>
> A sample for each of these files is available as part of the artifacts provided with the deployment package.

- When setting up data science jobs, create the following five jobs:
  - **Data Fetch Job**

    Set JOB_RUN_ENTRYPOINT to **entry_file.py**

    Job artifact: **data_fetch_from_os.zip**
  - **Preprocess Job**

    Set JOB_RUN_ENTRYPOINT to **preprocess_entry.py**

    Job artifact: **preprocess.zip**
  - **Train Job**

    Set JOB_RUN_ENTRYPOINT to **train_entry.py**

Copyright © 2026, Oracle and/or its affiliates.

Job artifact: **train.zip**

- **Artifact Job**

  Set JOB_RUN_ENTRYPOINT to **artifact_entry.py**

  Job artifact: **artifact.zip**

- **Deploy Job**

  Set JOB_RUN_ENTRYPOINT to **deployment_entry.py**

  Job artifact: **deploy.zip**

> **Note**
>
> Oracle provides all of these zip files as part of the artifacts given with the deployment package.

For non-OCI deployments: You only need to install the Helm charts for the training service as described below.

To install the training services using Helm charts:

1. Download the training utility service Helm charts from the deployment package.

   For more information, see "Downloading Packages for the Cloud Native Helm Charts and Docker Files" and "Setting Up Prerequisite Software and Tools".
2. Create a new key and generate an SSL certificate to run the data service over HTTPS.
3. (Optional) Install the Ingress NGINX controller:
   - Create a Kubernetes TLS secret using the certificate you generated and your private key:

     ```
     kubectl create secret tls secretName --cert=certificateName.crt --
     key=privateKeyName.key
     ```
   - Add the ingress-nginx Helm repository:

     ```
     helm repo add ingress-nginx
     ```
   - Update the Helm repository:

     ```
     helm repo update
     ```
   - Install the Ingress Controller:

     ```
     helm install ingress-nginx ingress-nginx/ingress-nginx --namespace
     nginxNamespace --create-namespace
     ```
   - Attach the service to the NGINX Controller:

     ```
     helm install ingress-nginx ingress-nginx/ingress-nginx --namespace
     nginxNamespace \
       --set controller.service.enableHttp=false \
       --set controller.service.enableHttps=true \
       --set controller.service.ports.https=443 \
       --set controller.service.nodePorts.https=31231 nodePort \
       --set controller.config.ssl-redirect=true \
       --set controller.config.force-ssl-redirect=true \
       --set controller.ingressClassResource.name=ingressClassName \
     ```


Copyright © 2026, Oracle and/or its affiliates.

```
--set controller.ingressClass=ingressClassName \
--create-namespace
```

4. Update the **values.yaml** file for your deployment:
   a. Set mount paths:
      - **mountPath.modelFilesHostPath**: Specify the path where model files are stored. For a single worker node, specify a path on that node. For multiple worker nodes, specify the NFS or OCI File Storage export path. Set this value even if the models are stored in OCI.
      - **mountPath.dataFilesHostPath**: Specify the path where data files are stored when **storage_type** is **pvc**. For a single worker node, specify a path on that node, such as **/mnt/data_service**. For multiple worker nodes, specify the NFS or OCI File Storage export path.
      - **mountPath.logFilesHostPath**: Specify the path where log files are stored. For multiple worker nodes, specify the NFS or OCI File Storage export path. If this value is not set, log files are stored in the directory specified by **mountPath.modelFilesHostPath**.
      - Set folder permissions:

        ```
        groupadd -g 10001 oracle
        useradd -mr -u 10001 -g oracle oracle
        chown 10001:10001 -R mountPath
        ```

   b. Set storage configuration:
      - **modelFilesStorage.storageType**: Set to **hostPath** for files on a single worker node, **nfs** for an NFS export shared across worker nodes, or **ociFss** for an OCI File Storage export shared across worker nodes. The default is **nfs**.
      - **modelFilesStorage.nfs.server**: For NFS storage, specify the DNS name or IP address of the NFS server. The server must export the path specified in **mountPath.modelFilesHostPath** and be reachable from the worker nodes.
      - **modelFilesStorage.nfs.existingClaim**: When upgrading a deployment that uses an existing model-files PVC, specify its current claim name.
      - **logFilesStorage.existingClaim** and **dataFilesStorage.existingClaim**: When upgrading a deployment that uses existing log-files or data-files PVCs, specify their current claim names.
      - **modelFilesStorage.ociFss.fileSystemOcid**: For OCI File Storage, specify the OCID of the existing file system.
      - **modelFilesStorage.ociFss.mountTargetIp**: For OCI File Storage, specify the mount target IP address.
   c. If you are not using NGINX Ingress, configure HTTPS:
      - **https.enabled**: Set to **true**.
      - **https.source**: Set to **hostPath** when Kubernetes Secrets are used for certificate ingestion, or to **chartFiles** to read the certificate and key from chart files.
      - **https.orchestratorServicePort**: Specify the port on which the service is exposed.
      - **https.path**: If **https.source** is **hostPath**, specify the host system path containing the certificate and key.

Copyright © 2026, Oracle and/or its affiliates.

- **https.certFile** and **https.keyFile**: If **https.source** is **hostPath**, specify the certificate and key file names in that path.
- **https.certPath** and **https.keyPath**: If **https.source** is **chartFiles**, specify the certificate and key files in the **helm/train-utility/files/certs/** directory.

d. Add **imageRepository** and image tags for **trainUtilityOrchestrator.image** and **trainUtilityProcessor.image**.

e. Under the **trainUtilityPredictor.configurableFiles** entry, specify the paths within the mount paths to place the following files:

- **preprocessScript**: Specify the path to the Python preprocessing script. Place it in **helm/train-utility/files/scripts/**.
- **datascienceConfig**: Specify the path to the configuration file containing the Data Science job details. Place it in **helm/train-utility/files/config/**.
- **labelFile**: Specify the path to the label file used to train a DNN, DNN_MH, or XGB model. Place it in **helm/train-utility/files/config/**.
- **logConfigFile**: Specify the path to the logging configuration file if you need to change the log level or format. Place it in **helm/train-utility/files/config/**.

Specify these paths relative to the mount path. If a configurable file is larger than 1 MB, place it under **mountPath.modelFilesHostPath** and specify its absolute path inside the container.

> **Note**
>
> Configurable sample files are available for each of the files mentioned above in the artifacts provided with the package: **sample_preprocess.py** (for preprocessScript), **oci_config** (for ociConfigFile), **config.json** (for datascienceConfig), **sample_label.csv** (for labelFile), and **logging_config.py** (for logConfigFile). You can configure these files as per your requirements.

f. If your deployment uses OCI services, configure OCI access:

- **ociConfig.enabled**: Set to **true** if the training service uses OCI Data Flow, Data Science, or Object Storage.
- **ociConfig.configPath** and **ociConfig.privateKeyPath**: Specify the relative paths of the OCI configuration file and private key, such as paths under **files/secrets/oci/**.

g. If you are using NGINX, set the NGINX Ingress values:

- **host**: Set to the host name where the service is deployed.
- **tlsSecretName**: Set to the name of the Kubernetes TLS secret created in step 3.
- **ingressClassName**: Set to your ingress class name.

h. Set **identityURI**, **issuer**, and **audience** to your IDCS configuration. Do one of the following:

- To use IDCS, set **trainUtilityOrchestrator.idcs.enabled** to **true**.
- To disable IDCS, set **trainUtilityOrchestrator.idcs.enabled** to **false**.

Copyright © 2026, Oracle and/or its affiliates.

5. Install the Helm charts:

```
helm upgrade --install training-utility-services helm/train-utility/ --
namespace=training-utility-service
```

Configuring and Using the Training Service

You interact with the training service using REST APIs, which enable integration with external systems.

For training service, you use the following REST service:

- **Train the Model (/utility/train**)

  Use this API to train a model based on the data acquired by the data service and the configured parameters.

For more information, see *REST API Reference for AI Services*.

Training a Multi-Head Offer Recommendation Model

Use a multi-head offer recommendation model when you want to generate multiple related recommendations for the same customer. For example, the model can recommend a charge or product offer and a discount offer.

To train a multi-head offer recommendation model:

1. Configure the data service to collect the required data. See "Configuring Multiple Data Sources".
2. Collect and persist the required data fragments.

   Ensure that the data fragments required for training are available in the configured storage location.
3. Create a label file.

   The label file must contain two or more columns that use the LABEL_ prefix. Each label column becomes an output head in the model. For example, LABEL_1 can contain charge or product offer labels and LABEL_2 can contain discount offer labels.
4. Use a preprocessing script to prepare the input data, select features, encode categorical values, or apply business rules before training.
5. Configure the storage location for the model, supporting artifacts, and metrics.
6. Configure OCI Data Science or MLflow, depending on your deployment environment.
7. Configure the training service helm chart.

   Specify the label file, preprocessing script, model storage, and logging configuration. For OCI deployments, also configure the required OCI Data Science settings.
8. Submit a training request that specifies the dnn_mh algorithm.

   The training service validates the label columns and input data before training starts.
9. Verify that the training service created the model and supporting artifacts.

   The training service trains a multi-head model, evaluates each output head, and stores the model in ONNX format. Supporting artifacts can include label encoders, feature-column metadata, and metrics files.

Copyright © 2026, Oracle and/or its affiliates.

10. Verify that the model is available in the configured model-management system.

    The training service publishes the model and metrics to OCI Data Science or MLflow, depending on the deployment configuration.

> **Note**
>
> Do not configure the single-head DNN algorithm and the multi-head DNN algorithm in the same training request.

Training an Anomaly Detection Model

Use an anomaly detection model to identify unusual behavior in time-series operational metrics. The supplied sample use case uses an XGBoost model.

To train an anomaly detection model:

1. Configure the data service to collect the required Prometheus metrics. See "Collecting Metrics for Anomaly Detection".
2. Collect and persist the historical metric data.

   Ensure that the metric data required for training is available in the configured storage location.
3. Create a label file.

   The label file must contain a label for each record in the collected metric data. The supplied sample use case uses 0 for normal metrics and 1 for potential outage detection.

   You can add labels, such as 2, 3, and 4, when your use case identifies different anomaly types. Define the meaning of each label in the label file.
4. Use a preprocessing script to prepare the input data, select the required metric features, and associate the metric data with the labels before training.

   Ensure that the preprocessing script prepares the features in the format expected by the model. The same features must be available when you request a prediction.
5. Configure the storage location for the trained model, metadata, supporting artifacts, and training metrics.
6. Configure OCI Data Science or MLflow, depending on your deployment environment.
7. Configure the training service helm chart.

   Specify the label file, preprocessing script, model storage location, and logging configuration. For OCI deployments, also configure the required OCI Data Science settings.
8. Submit a training request that specifies the algorithm configured for the model.

   The training service validates the input data and configured parameters before training starts.
9. Verify that the training service created the model and supporting artifacts.

   The training service trains the model and stores the metadata required by the prediction service. The metadata identifies the features that the prediction service uses to prepare the metric data.
10. Verify that the model is available in the configured model-management system.

Copyright © 2026, Oracle and/or its affiliates.

The training service publishes the model and training metrics to OCI Data Science or MLflow, depending on the deployment configuration.

Copyright © 2026, Oracle and/or its affiliates.

6

Deploying and Configuring the Prediction Service

Learn how to deploy and configure the prediction service for Oracle Monetization Suite in both Oracle Cloud Infrastructure (OCI) and non-OCI environments.

Topics in this document:

- Deploying the Prediction Service
- Configuring and Using the Prediction Service
- Configuring the MCP Interface
- Predicting Time-Series Anomalies

Deploying the Prediction Service

You can connect to a model on Oracle Cloud Infrastructure or in a non-OCI environment and deploy the prediction service independently, depending on your requirements. For this, you need to perform the installation steps described in "Overview of Installation Tasks", and then install the prediction service Helm charts by following the instructions below.

To install and deploy the prediction service using Helm charts:

1. Download the prediction service Helm charts from the deployment package.

   For more information, see "Downloading Packages for the Cloud Native Helm Charts and Docker Files" and "Setting Up Prerequisite Software and Tools".
2. Create a new key and generate an SSL certificate to run the data service over HTTPS.
3. (Optional) Install the Ingress NGINX controller:
   - Create a Kubernetes TLS Secret using the certificate you generated and your private key:

     ```
     kubectl create secret tls secretName --cert=certificateName.crt --key=privateKeyName.key
     ```
   - Add the ingress-nginx Helm repository:

     ```
     helm repo add ingress-nginx
     ```
   - Update the Helm repository:

     ```
     helm repo update
     ```
   - Install the Ingress Controller:

     ```
     helm install ingress-nginx ingress-nginx/ingress-nginx --namespace nginxNamespace --create-namespace
     ```
   - Attach the service to the NGINX Controller:

     ```
     helm install ingress-nginx ingress-nginx/ingress-nginx --namespace nginxNamespace \
       --set controller.service.enableHttp=false \
     ```


Copyright © 2026, Oracle and/or its affiliates.

```
--set controller.service.enableHttps=true \
--set controller.service.ports.https=443 \
--set controller.service.nodePorts.https=31231 nodePort \
--set controller.config.ssl-redirect=true \
--set controller.config.force-ssl-redirect=true \
--set controller.ingressClassResource.name=ingressClassName \
--set controller.ingressClass=ingressClassName \
--create-namespace
```

4. Update the **values.yaml** file for your deployment:
   a. Set mount paths:
      - **mountPath.modelFilesHostPath**: Specify the path where model files are stored. For a single worker node, specify a path on that node. For multiple worker nodes, specify the NFS or OCI File Storage export path. Set this value even if the models are stored in OCI.
      - **mountPath.logFilesHostPath**: Specify the path where log files are stored. For multiple worker nodes, specify the NFS or OCI File Storage export path. If this value is not set, log files are stored in the directory specified by **mountPath.modelFilesHostPath**.
      - Set folder permissions:

        ```
        groupadd -g 10001 oracle
        useradd -mr -u 10001 -g oracle oracle
        chown 10001:10001 -R mountPath
        ```

   b. Set storage configuration:
      - **modelFilesStorage.storageType**: Set to **hostPath** for files on a single worker node, **nfs** for an NFS export shared across worker nodes, or **ociFss** for an OCI File Storage export shared across worker nodes. The default is **nfs**.
      - **modelFilesStorage.nfs.server**: For NFS storage, specify the DNS name or IP address of the NFS server. The server must export the path specified in **mountPath.modelFilesHostPath** and be reachable from the worker nodes.
      - **modelFilesStorage.nfs.existingClaim**: When upgrading a deployment that uses an existing model-files PVC, specify its current claim name.
      - **logFilesStorage.existingClaim**: When upgrading a deployment that uses an existing log-files PVC, specify its current claim name.
      - **modelFilesStorage.ociFss.fileSystemOcid**: For OCI File Storage, specify the OCID of the existing file system.
      - **modelFilesStorage.ociFss.mountTargetIp**: For OCI File Storage, specify the mount target IP address.
   c. If you are not using NGINX Ingress, configure HTTPS:
      - **https.enabled**: Set to **true**.
      - **https.source**: Set to **hostPath** when Kubernetes Secrets are used for certificate ingestion, or to **chartFiles** to read the certificate and key from chart files.
      - **https.orchestratorServicePort**: Specify the port on which the service is exposed.
      - **https.path**: If **https.source** is **hostPath**, specify the host system path containing the certificate and key.

Copyright © 2026, Oracle and/or its affiliates.

- **https.certFile** and **https.keyFile**: If **https.source** is **hostPath**, specify the certificate and key file names in that path.
- **https.certPath** and **https.keyPath**: If **https.source** is **chartFiles**, specify the certificate and key files in the **helm/predict-service/files/certs/** directory.

d. Add **imageRepository** and image tags for **predictionOrchestrator.image** and **predictionPredictor.image**.

e. Under **predictionPredictor.configurableFiles**, specify the following values.

- **modelPath**: Specify the path for the base folder where models are stored. Supported: PVC, OCI bucket (must include oci://), or Data Science URI (must include https://).
- **modelNames**: Specify the file path for the list of models to be loaded when the service starts. Update and restart the service if new models are added.
  - For the PVC path, follow the {Stage}@{Model_Name} format. The files with '@' are stored in MLFlow. The remaining files are required supporting files.
  - For the models in OCI object storage, follow the {Name}@{OCI_Path} format. For KNN or Cosine models, also include the required mapper file paths from Object Storage.
  - For OCI Data Science model deployments, follow the {model_name}|{algorithm_name}@{Predict_URI} format. The model_name must match the name used during training. For example, if the trained models are named **recommend_knn** and **recommend_dnn**, entries can follow this pattern:

    ```
    ["recommend_knn|knn@{Predict_URI}", "recommend_dnn|dnn@{Predict_URI}"]
    ```

  - For an XGBoost model stored in OCI Object Storage, include its metadata file in **modelNames**.
- **preprocessScript**: Specify the path to the preprocessing script used for prediction. Place it in **helm/prediction/files/scripts/**.
- **logConfigFile**: Specify the path to the logging configuration file if you need to change the log level or format. Place it in **helm/prediction/files/config/**.
- **mcpPluginManifest**: Specify the manifest for additional MCP server tools. Place it in **helm/prediction/files/plugin/**, along with the **custom_tools.py** script.

> **Note**
>
> Configurable sample files are available for each of the files mentioned above in the artifacts provided with the package: **oci_config** (for ociConfigFile), **sample_predict_preprocess.py** (for preprocessScript), **logging_config.py** (for logConfigFile), **example-tools.json** (for mcpPluginManifest), and **sample_echo_context.py** (for the custom_tools.py script). You can configure these files as per your requirements.

f. If your deployment uses OCI services, configure OCI access:

- **ociConfig.enabled**: Set to **true** if the prediction service uses OCI Data Flow, Data Science, or Object Storage.
- **ociConfig.configPath** and **ociConfig.privateKeyPath**: Specify the relative paths of the OCI configuration file and private key, such as paths under **files/secrets/oci/**.

Copyright © 2026, Oracle and/or its affiliates.

g. If you are using NGINX, set the NGINX Ingress values:
   - **host**: Set to the host name where the service is deployed.
   - **tlsSecretName**: Set to the name of the Kubernetes TLS secret created in step 3.
   - **ingressClassName**: Set to your ingress class name.

h. Set **identityURI**, **issuer**, and **audience** to your IDCS configuration. Do one of the following:
   - To use IDCS, set **predictionOrchestrator.idcs.enabled** to **true**.
   - To disable IDCS, set **predictionOrchestrator.idcs.enabled** to **false**.

i. Configure the MCP server, if required:
   - **predictionOrchestrator.mcp.enabled**: Set to **true** to start the MCP server.
   - **predictionOrchestrator.mcp.endpointPath**: Set to the required MCP endpoint pattern.

5. Install the Helm charts:

```
helm upgrade --install predict-services helm/prediction/ --namespace=predict-service
```

Configuring and Using the Prediction Service

You can use and configure the prediction service using its REST API. The prediction service accepts API calls from other systems for inference and recommendation.

For the prediction service, you use the following REST service:

- **Get Predictions (/utility/predict**)

  Use this API to generate predictions or recommendations for specific input data after models are trained and deployed.

The prediction service also provides a Model Context Protocol (MCP) endpoint at **/monetization/nbo/v1/mcp**. This endpoint is configurable. MCP clients can use this endpoint to run predictions, validate prediction requests, and list supported prediction algorithms. Send MCP requests to this endpoint using HTTP POST and the JSON-RPC 2.0 format.

> **Note**
>
> Interactive REST API documentation is available through the Swagger UI at the **/openapi/ui** endpoint.

Configuring the MCP Interface

The prediction service supports a Model Context Protocol (MCP) interface for AI agents and MCP clients. The MCP interface enables AI agents and MCP clients to request predictions from the prediction service by using a standard tool-based interface.

Use the MCP interface when you want an AI agent to request predictions without directly accessing the prediction service implementation, model artifacts, or underlying data sources. The MCP interface provides a controlled integration point for validating requests, routing them to the prediction service, and returning structured prediction results.

To configure the MCP interface:

Copyright © 2026, Oracle and/or its affiliates.

1. In the prediction service helm chart, specify the model path and the list of approved model artifacts to load when the prediction service starts.
2. Do the following to configure the settings for MCP:
   - Enable the MCP server.

     Set **predictionOrchestrator.mcp.enabled** to **true** in the Helm values file.
   - Specify the public MCP endpoint path.

     Set the required public endpoint path for the MCP server. For example, set **predictionOrchestrator.mcp.endpointPath** to the required path in the Helm values file. Ensure that the path does not conflict with existing Prediction Service REST API paths.
   - Specify the prediction preprocessing script.

     Add the prediction preprocessing script to the Prediction Service artifacts directory. Set the script path in **predictionPredictor.configurableFiles.preprocessScript** in the Helm values file.
   - Specify the logging configuration.

     Add the logging configuration file to the Prediction Service artifacts directory. Set the configuration-file path in **predictionPredictor.configurableFiles.logConfigFile** in the Helm values file.
   - (Optional) Configure an MCP plugin manifest and custom tool implementation.

     Add the MCP plugin manifest and custom tool implementation files to the Prediction Service artifacts directory. Set the plugin manifest path in **predictionPredictor.configurableFiles.mcpPluginManifest** in the Helm values file.
   - Configure ingress, TLS, authentication, and service-routing settings.

     Configure the ingress host, TLS secret, and ingress class in the Helm values file. Configure the required authentication settings, such as IDCS settings, and route requests from the MCP endpoint to the Prediction Service orchestrator.
3. Deploy or restart the prediction service after making the updates.

   The prediction service loads the configured model artifacts when it starts.
4. Verify that the public MCP endpoint is available.

   The Helidon orchestrator exposes the public MCP interface.
5. Verify that external clients use the public MCP endpoint.

   The internal Python prediction runtime can provide internal MCP capabilities. External clients must not access the internal Python MCP endpoint directly.
6. Use the configured health checks, logs, metrics, and tracing information to monitor service availability, request processing, and errors.

Implementing Siebel Next Best Offer Recommendations

After you configure the MCP interface, you can use it to request predictions or recommendations from an AI Agent or other MCP client. In this case, you can use it to request next best offer recommendations from a Siebel AI Agent.

Before sending a request, ensure that the prediction service is running, has loaded the approved multi-head model, and can access the required cached customer data.

To request an offer recommendation:

Copyright © 2026, Oracle and/or its affiliates.

1. Send an MCP request from the AI Agent to the public MCP endpoint.
2. Specify the supported customer identifier in the request.
3. The MCP interface validates the request and sends it to the prediction service.
4. The prediction service retrieves the required customer data, runs the configured model, and returns the recommendation results to the MCP interface.
5. Verify that the MCP response contains the expected recommendation outputs.
6. Review the configured logs, metrics, and tracing information to monitor the request and troubleshoot errors.

Predicting Time-Series Anomalies

Prediction services can help you identify any anomalies in the regular pattern or behavior. You can use an anomaly detection model to classify current operational metrics as normal or anomalous. The supplied sample use case uses an XGBoost model.

Before you request an anomaly detection prediction, ensure that the prediction service is configured with the trained model, model metadata, and Prometheus connection details.

To request an anomaly detection prediction:

1. In the prediction request, specify the input fields as required.

   The sample implementation uses the start time, end time, and time step to specify the period and interval for retrieving metrics. You can customize the JSON input fields to meet your use case requirements.
2. Submit the prediction request through the configured prediction service or MCP interface.

   The prediction service retrieves the required metrics from Prometheus and prepares the input data by using the feature definitions in the model metadata.
3. Review the classification returned for the latest timestamp in the requested time window.

   The supplied sample use case returns 0 for normal metrics and 1 for anomalous metrics.

You can interpret the returned value according to the labels defined for the deployed model. If the model uses additional labels, use the label definitions configured for that model to identify the anomaly type. Verify that all required metrics are available in Prometheus and that they match the feature definitions in the model metadata.

> **Note**
>
> If a required metric is unavailable or its feature definition differs from the training data, the prediction service might be unable to prepare valid input for the model.

You can also submit an anomaly detection prediction request through the configured MCP interface. For information about configuring the MCP interface, see "Configuring the MCP Interface".

Copyright © 2026, Oracle and/or its affiliates.

7

Deploying and Configuring the Billing Care AI Agent

Learn how to deploy, configure, and use the Billing Care AI Agent using OCI hosted deployment or Kubernetes, with Oracle Cloud Infrastructure (OCI) Generative AI.

Topics in this document:

- Deploying and Configuring the Billing Care AI Agent
- Using the Billing Care AI Agent

Deploying the Billing Care AI Agent

You can deploy the Billing Care AI Agent in either of the following ways:

- Deploying in a Kubernetes Environment
- Deploying using OCI Generative AI hosted deployment

Deploying the Billing Care AI Agent in Kubernetes

To deploy the Billing Care AI Agent in Kubernetes:

1. In your **values.yaml** file, set the following keys:
   - **enabled**: Set to **true** for a standard deployment.
   - **imageRepository**: Specify the container registry and repository prefix, such as `<registry>/<repository>`. This value is combined with **billingcareAgent.container.image**.
   - **imagePullSecrets**: Specify the name of an existing Kubernetes secret containing credentials for accessing your private container registry. Leave empty if a secret is not required.
   - **imagePullPolicy**: Specify **IfNotPresent** for an immutable release tag or **Always** when validating a mutable tag.
   - **billingcareAgent.container.image**: Specify the agent image name and immutable release tag, in this format - `communications/ai/agents/:<release-tag>`.
2. Set the following keys under **billingcareAgent.env**:
   - **BILLING_CARE_MCP_URL**: Specify the Billing Care MCP streamable HTTP endpoint. Use HTTPS. If the runtime does not trust the server certificate chain, configure **mcpTls** as described in the MCP certificate configuration step.
   - **OCI_GENAI_MODEL_ID**: Specify the OCI Generative AI model identifier.
   - **OCI_GENAI_SERVICE_ENDPOINT**: Specify the OCI Generative AI inference endpoint for the region.
   - **OCI_COMPARTMENT_ID**: Specify the OCI compartment OCID authorized to use the model.

Copyright © 2026, Oracle and/or its affiliates.

- **OCI_AUTH_TYPE**: Specify **RESOURCE_PRINCIPAL** for resource principal authentication or **API_KEY** for API key authentication. For API key authentication, configure the secret mount under **ociConfig** as described in the OCI authentication step.
- **OCI_AUTH_PROFILE**: Optionally, specify the OCI configuration profile to use for API key authentication.
- **OCI_AUTH_FILE_LOCATION**: Normally, leave empty. When **ociConfig.enabled** is **true**, the chart sets this value to the mounted OCI configuration file path.
- **BILLING_MCP_HTTP_TIMEOUT_SECONDS**: Specify the MCP request timeout, in seconds. The default is **30**.
- **MCP_MAX_RETRIES**: Specify the maximum number of retry attempts for eligible MCP operations. The default is **2**.
- **SESSION_MEMORY_DIR**: Specify an absolute writable path for mounted session storage. Set to **none** to retain session context in process memory only.
- **SESSION_MEMORY_TTL_MINUTES**: Specify the idle-session retention period, in minutes. The default is **60**.
- **SESSION_MEMORY_MAX_MESSAGES**: Specify the maximum number of optimized messages retained per session. The default is **100**.
- **SESSION_MEMORY_VERBATIM_TURNS**: Specify the number of recent conversation turns retained verbatim before older history is compacted. The default is **5**.
- **SESSION_ACCOUNT_HISTORY_LIMIT**: Specify the maximum number of accounts retained in a session. Set to **0** to retain all account references for that session.
- **BILL_LOOKUP_MAX_COUNT**: Specify the maximum number of bill records considered for a lookup. The default is **50**.
- **MAX_FOLLOW_UP_QUESTIONS**: Specify the maximum number of follow-up questions generated for an agent response. The default is **3**.
- **LOG_DIR**: Specify the log directory in the container. This value must match **billingcareAgent.logs.mountPath**.
- **LOG_FILE**: Specify the log filename.
- **LOG_CONFIG_FILE**: Optionally, specify the path to the YAML logging configuration file in the container. Configure the logging file mount as described in the logging configuration step.

Do not include tokens, passwords, private keys, or other secrets under **billingcareAgent.env**.

3. Configure OCI authentication by doing one of the following:
   - For resource-principal authentication, set **billingcareAgent.env.OCI_AUTH_TYPE** to **RESOURCE_PRINCIPAL** and leave **ociConfig.enabled** set to **false**.
   - For API-key authentication, set **billingcareAgent.env.OCI_AUTH_TYPE** to **API_KEY** and **ociConfig.enabled** to **true**. Set **ociConfig.secretName** to the name of an existing secret containing the OCI configuration and credential material. Set **ociConfig.configFile** and **ociConfig.mountPath** to the configuration filename and mount directory.
4. Configure session storage by setting the following keys:
   - **billingcareAgent.env.SESSION_MEMORY_DIR**: Specify an absolute writable path for mounted session storage. Set to **none** to retain session context in process memory only.

Copyright © 2026, Oracle and/or its affiliates.

- **persistence.enabled**: Set to **true** to retain session memory across pod restarts.
- **persistence.storageClassName**: Specify an existing StorageClass, or leave empty to use the cluster default.
- **persistence.accessMode**: Specify the volume access mode.
- **persistence.size**: Specify the storage size.
- **billingcareAgent.env.SESSION_MEMORY_TTL_MINUTES**: Specify the idle-session retention period, in minutes.
- **billingcareAgent.env.SESSION_MEMORY_MAX_MESSAGES**: Specify the maximum number of messages retained per session.

If the chart must create a StorageClass for session memory, configure the following keys:

- **storageClass.enabled**: Set to **true** to create a StorageClass. The default is **false**.
- **storageClass.name**: Specify the StorageClass name. Leave empty to use **<fullname>-session-memory**.
- **storageClass.provisioner**: Specify the installed CSI provisioner. The default is **blockvolume.csi.oraclecloud.com**.
- **storageClass.reclaimPolicy**: Specify **Retain** to preserve the backing volume after PVC deletion or **Delete** to remove it. The default is **Retain**.
- **storageClass.volumeBindingMode**: Specify **Immediate** or **WaitForFirstConsumer**. The default is **WaitForFirstConsumer**, which defers provisioning until pod scheduling.
- **storageClass.allowVolumeExpansion**: Set to true to allow volume expansion when supported by the CSI driver. The default is **true**.
- **storageClass.annotations**: Specify annotations for the StorageClass, if required. The default is **{}**.
- **storageClass.parameters**: Specify provisioning parameters supported by the selected provisioner. The default is **attachment-type: paravirtualized**.

The chart creates the StorageClass only when **storageClass.enabled** and **persistence.enabled** are both **true** and **persistence.storageClassName** is empty. The Helm service account must have permission to create StorageClass resources.

For multiple replicas, use persistent or external session storage, or session-aware routing. Process-local session memory alone is not sufficient.

5. Configure HTTPS access by doing one of the following:
   - For NGINX Ingress, set **nginx.enabled** to **true**. Set **nginx.ingressClassName** to the ingress class, **nginx.host** to the agent hostname, and **nginx.tlsSecretName** to the name of an existing TLS secret.
   - (Recommended) For HTTPS termination directly in the agent, set **nginx.enabled** to **false** and configure the following keys:
     - **https.enabled**: Set to **true**.
     - **https.source**: Specify **chartFiles** or **hostPath**. The default is **chartFiles**.
     - **https.mountPath**: Specify the read-only certificate directory in the container.
     - **https.certFile**: Specify the certificate-chain filename under the mount directory.
     - **https.keyFile**: Specify the private-key filename under the mount directory.
     - **https.containerPort**: Specify the HTTPS listener port in the container.
     - **https.servicePort**: Specify the HTTPS port exposed by the Kubernetes Service.

Copyright © 2026, Oracle and/or its affiliates.

  - **https.hostPath.path**: For **hostPath**, specify the absolute node directory containing the certificate and private key.
  - **https.hostPath.type**: For **hostPath**, specify **Directory**.
  - **https.chartFiles.certPath**: For **chartFiles**, specify the chart-relative certificate-chain path in the **helm/billingcare-agent/files/certs/** directory.
  - **https.chartFiles.keyPath**: For **chartFiles**, specify the chart-relative private-key path in the **helm/billingcare-agent/files/certs/** directory.

  For direct HTTPS, the certificate must include the agent hostname in its subject alternative name. If you use **hostPath**, the certificate files must exist on every node on which the pod can run and must be readable by UID/GID 10001.

6. If the HTTPS MCP endpoint uses a certificate authority that the runtime does not trust, configure the following keys:
   - **mcpTls.enabled**: Set to **true** to mount a CA bundle for outbound MCP certificate validation. The default is **false**.
   - **mcpTls.source**: Specify **chartFiles** or **hostPath**. The default is **chartFiles**.
   - **mcpTls.mountPath**: Specify the read-only CA directory in the container.
   - **mcpTls.caFile**: Specify the CA-bundle filename under the mount directory.
   - **mcpTls.hostPath.path**: For **hostPath**, specify the absolute node directory containing the CA bundle.
   - **mcpTls.hostPath.type**: For **hostPath**, specify **Directory**.
   - **mcpTls.chartFiles.caPath**: For **chartFiles**, specify the chart-relative CA-bundle path in the **helm/billingcare-agent/files/certs/mcp** directory.

   Do not disable certificate validation. The inbound agent certificate and outbound MCP CA bundle are configured separately.
7. Configure MCP authentication:
   - For caller-token forwarding, leave **mcpOAuth.enabled** set to **false**.
   - For an approved service-to-service token configuration, set **mcpOAuth.enabled** to **true**. Set **mcpOAuth.secretName** to the name of an existing secret and **mcpOAuth.secretKey** to the key containing the raw token without the `Bearer` prefix.
8. If the agent validates incoming bearer tokens, set **billingcareAgent.idcs.enabled** to **true** and configure the following keys under **billingcareAgent.idcs**:
   - **identityURI**: Specify the identity-provider URI.
   - **issuer**: Specify the token issuer.
   - **audience**: Specify the token audience.
   - **jwksURI**: Specify the signing-key URI.
9. Configure the workload settings:
   - Set the CPU and memory requests and limits under **billingcareAgent.container.resources**.
   - Set the service type and port under **billingcareAgent.service**.
   - Configure autoscaling under **billingcareAgent.hpa**. If autoscaling is **disabled**, set the replica count by using **billingcareAgent.replicas**.

Copyright © 2026, Oracle and/or its affiliates.

10. Configure logging under **billingcareAgent.logs**. Ensure that **billingcareAgent.env.LOG_DIR** matches **billingcareAgent.logs.mountPath** and that the log directory is writable by UID/GID 10001.

    To use persistent log storage, set **billingcareAgent.logs.enabled** to **true**, set **billingcareAgent.logs.hostPath.enabled** to **false**, and configure the following keys:

    - **logPersistence.enabled**: Set to **true** to create a log PVC. The default is **false**.
    - **logPersistence.accessMode**: Specify the volume access mode supported by the storage provider. The default is **ReadWriteOnce**.
    - **logPersistence.size**: Specify the storage size. The default is **1Gi**.
    - **logPersistence.annotations**: Specify annotations for the log PVC, if required. The default is **{}**.

    The log PVC uses the cluster default StorageClass. No separate StorageClass setting is available for logs. When **billingcareAgent.logs.hostPath.enabled** is **true**, node-local storage takes precedence over **logPersistence**. When log storage is enabled and both **billingcareAgent.logs.hostPath.enabled** and **logPersistence.enabled** are **false**, the chart uses ephemeral **emptyDir** storage.

    > **Note**
    >
    > Do not include passwords, OCI private keys, OAuth tokens, or certificates directly in the values file. Use the applicable Secret configuration.

11. Validate the Helm chart by running this command:

    ```
    helm lint helm/billingcare-agent
    ```

12. Generate the Kubernetes resource definitions by using helm template with **values.yaml**. Review the generated deployment, service, Ingress, persistent volume claim, certificate mounts, and environment variables.
13. Deploy the Helm chart with **values.yaml**.

Perform checks to ensure that everything is correct leading to a successful deployment:

- Confirm that the pod starts and that **/health** returns an **OK** status.
- Confirm that **/ready** returns a ready status and that the MCP URL is configured.
- Submit a simple bill request to **/invoke/chat** with a valid authorized token.
- Verify that session storage, logging, and MCP certificate validation work as configured.
- Confirm that logs do not contain tokens, credentials, full invoices, payment details, or unnecessary personally identifiable information.

Deploying the Billing Care AI Agent by Using OCI Generative AI Hosted Deployment

To deploy the Billing Care AI Agent by using OCI Generative AI Hosted Deployment:

1. In the OCI Console, open **Generative AI** and create a hosted application. Specify the application name, description, and compartment.
2. Configure the minimum and maximum replica counts, scaling metric, and CPU threshold.

Copyright © 2026, Oracle and/or its affiliates.

For multiple replicas, configure session storage or routing so that sessions do not depend on process-local memory.

3. Enable managed storage only if the agent requires the available PostgreSQL, cache, or Autonomous Database services.
4. Add the required application environment variables, including:
   - **BILLING_CARE_MCP_URL**: Specify the Billing Care MCP streamable HTTP endpoint.
   - **OCI_GENAI_MODEL_ID**: Specify the OCI Generative AI model identifier.
   - **OCI_GENAI_SERVICE_ENDPOINT**: Specify the OCI Generative AI inference endpoint.
   - **OCI_COMPARTMENT_ID**: Specify the OCI compartment OCID.
   - **OCI_AUTH_TYPE**: Specify **RESOURCE_PRINCIPAL** when using resource principal authentication.
   - **IDCS_ENABLED**: Set to **true** when the agent validates incoming bearer tokens. Set to false when the hosted deployment platform performs inbound OAuth validation.
   - **IDCS_ISSUER**: Specify the expected token issuer when agent-level token validation is enabled.
   - **IDCS_AUDIENCE**: Specify the expected token audience when agent-level token validation is enabled.
   - **IDCS_JWKS_URI**: Specify the identity provider's JSON Web Key Set (JWKS) URI used to validate token signatures.
   - **PRINT_LOGS**: Set to **true** to enable log output.
   - **MCP_TLS_CA_CERT_BASE64**: If the Billing Care MCP endpoint uses a certificate authority that the runtime does not trust, supply the base64-encoded CA certificate through a Vault environment variable.
   - **NO_PROXY**: If your environment uses a proxy, specify the hosts or domains that the agent must access directly.

   Configure the session-memory, logging, and identity settings for the environment. Use Vault environment variables for secrets. Do not enter credentials, bearer tokens, private keys, or CA material as plain-text environment variables.
5. Configure network access:
   - For a private enterprise integration, select Custom (user-managed) networking and specify the VCN, subnet, and network security group.
   - Ensure that DNS, route tables, and network security rules allow outbound HTTPS access to OCI Generative AI and the Billing Care MCP server.
   - Select a private endpoint when the application must be reachable only within the VCN. Select a public endpoint only when external access and the required security controls have been approved.
6. Configure application authentication for the environment.

   The available options include identity-domain bearer-token authentication, OCI IAM request-signature authentication, and no application-layer authentication.
7. Enable resource logging and select the log group and log category.
8. After creating the application, open the configuration tab and verify the environment variables.

Copyright © 2026, Oracle and/or its affiliates.

9. Open the **Deployment** tab, review the autoscaling settings, and select **Create deployment**. Select the Billing Care AI Agent image and an immutable release tag from the approved container repository.
10. Create the deployment and monitor its work request until it succeeds.
11. Confirm that the deployment is active and that the application logs show a successful web-server startup.

Perform checks to ensure that everything is correct leading to a successful deployment:

- Call **/health** and **/ready** through the application endpoint.
- Submit a simple bill request to **/invoke/chat** with a valid authorized token. Confirm that the agent can reach OCI Generative AI and the Billing Care MCP endpoint.
- Verify session behavior and error handling.
- Confirm that logs do not contain sensitive information.

Record the deployed image tag or digest, application name, endpoint type, and environment-specific change record.

Using the Billing Care AI Agent

To use the Billing Care AI Agent:

1. Send a request to the agent health endpoint:

   ```
   curl https://<agent-host>/health
   ```

   where <agent-host> is the hostname of the Billing Care AI Agent.

   Confirm that the endpoint returns an OK status.
2. Send a request to the readiness endpoint:

   ```
   curl https://<agent-host>/ready
   ```

   Confirm that the endpoint returns a ready status.
3. Log in to Billing Care using an account that has access to the AI agent.

   > **Note**
   >
   > For the AI Agent in Billing Care, the chat UI is disabled by default. So, ensure that the chat UI is enabled along with the other required tools and resources.

4. Open a customer account that contains a bill.
5. Open the AI Agent chat and enter a bill inquiry. For example:

   Explain the charges on my latest bill.

   The agent retrieves the bill information and displays a response.

Copyright © 2026, Oracle and/or its affiliates.