Cost Management Service 1-latest

Integrating Oracle Cloud data into cost management

Learn how to add and configure your Oracle Cloud integration

Last Updated: 2024-10-29

Cost Management Service 1-latest Integrating Oracle Cloud data into cost management

Learn how to add and configure your Oracle Cloud integration

Legal Notice

Abstract

Learn how to add an Oracle Cloud integration to cost management. Cost management is part of the Red Hat Insights portfolio of services. The Red Hat Insights suite of advanced analytical tools helps you to identify and prioritize impacts on your operations, security, and business.

Table of Contents

CHAPTER 1. CREATING AN ORACLE CLOUD INTEGRATION 3
1.1. ADDING AN ORACLE CLOUD INFRASTRUCTURE ACCOUNT AND NAMING YOUR INTEGRATION 3
1.2. COLLECTING AND STORING YOUR GLOBAL COMPARTMENT-ID 3
1.3. CREATING A POLICY TO CREATE COST AND USAGE REPORTS 4
1.4. CREATING A BUCKET FOR ACCESSIBLE COST AND USAGE REPORTS 5
1.5. REPLICATING REPORTS TO A BUCKET 6
1.6. CREATING A BUCKET POLICY TO GRANT READ ACCESS AND FINAL STEPS 10

CHAPTER 2. NEXT STEPS FOR MANAGING YOUR COSTS 12
2.1. LIMITING ACCESS TO COST MANAGEMENT RESOURCES 12
2.2. CONFIGURING TAGGING FOR YOUR INTEGRATIONS 12
2.3. CONFIGURING COST MODELS TO ACCURATELY REPORT COSTS 13
2.4. VISUALIZING YOUR COSTS WITH COST EXPLORER 13

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION 14

CHAPTER 1. CREATING AN ORACLE CLOUD INTEGRATION

To add an Oracle Cloud account to cost management, you must add your Oracle Cloud account as an integration from the Red Hat Hybrid Cloud Console user interface and configure Oracle Cloud to provide metrics. After you add your Oracle Cloud account to cost management as a data integration, you must configure a function script to copy the cost and usage reports to a bucket that cost management can access.

Prerequisites

- Red Hat account user with Cloud Administrator entitlements
- Access to Oracle Cloud Console with access to the compartment you want to add to cost management
- A service on Oracle Cloud generating service usage

As you will complete some of the following steps in Oracle Cloud, and some steps in the Red Hat Hybrid Cloud Console, log in to both applications and keep them open in a web browser. To begin, add your Oracle Cloud integration to cost management from the Integrations page using the **Add a cloud integration** dialog.

1.1. ADDING AN ORACLE CLOUD INFRASTRUCTURE ACCOUNT AND NAMING YOUR INTEGRATION

Add your Oracle Cloud account as a integration. After adding an Oracle Cloud integration, the cost management application processes the cost and usage data from your Oracle Cloud account and makes it viewable.

Procedure

1. From Red Hat Hybrid Cloud Console , click **Settings Menu**  > **Integrations**.
2. On the **Settings** page, click **Integrations**.
3. In the **Cloud** tab, click **Add integration**.
4. In the **Add a cloud integration** wizard, select **Oracle Cloud Infrastructure** as the integration type. Click **Next**.
5. Enter a name for your integration and click **Next**.
6. In the **Select application** step, select **Cost management**. Click **Next**.

1.2. COLLECTING AND STORING YOUR GLOBAL COMPARTMENT-ID

Continue in the **Add a cloud integration** wizard by collecting your global **compartment-id**, which is also known as your **tenant-id** in Microsoft Azure, so cost management can access your Oracle Cloud compartment.

Procedure

1. In the **Add a cloud integration** wizard, on the **Global compartment-id** step, copy the command in step one **oci iam compartment list**.
2. In a new tab, log in to your Oracle Cloud account.
3. In the menu bar, click **Developer tools → Cloud Shell**.
4. Paste the command you copied from the **Add a cloud integration** wizard in the **Cloud Shell** window.
5. In the response, copy the value pair for the **compartment-id** key. In the following example, the ID starts with **ocid1.tenancy.oc1**.

   Example response

```
{
  "data": [
    {
      "compartment-id":
"ocid1.tenancy.oc1..00000000000000000000000000000000000000000000000000",
      "defined-tags": {
        "Oracle-Tags": {
            ...
        }
      },
      ...
    }
  ]
}
```

6. Return to the **Global compartment-id** step in the **Add a cloud integration** wizard, and paste your **tenant-id** in the **Global compartment-id** field.
7. Click **Next**.

1.3. CREATING A POLICY TO CREATE COST AND USAGE REPORTS

Continue in the **Add a cloud integration** wizard by creating a custom policy and compartment for Oracle Cloud to create and store cost and usage reports.

Procedure

1. In the **Add a cloud integration** wizard, on the **Create new policy and compartment** page, copy the **oci iam policy create** command.
2. Paste the command that you copied into the Cloud Shell in your Oracle Cloud tab to create a cost and usage reports policy. You can also add a policy description.
3. Return to the **Create new policy and compartment** step in the **Add a cloud integration** wizard and copy the **oci iam compartment create** command.
4. Paste the command that you copied into the Cloud Shell in your Oracle Cloud tab to create a cost management compartment.

5. In the response, copy the value for the **id** key. In the following example, copy the id that includes **ocid1.compartment.oc1**.

   Example response


```
{
  "data": [
    {
      "compartment-id": "tenant-id",
      "defined-tags": {
        "Oracle-Tags": {
            ...
        }
      },
      "description": "Cost management compartment for cost and usage data",
      "freeform-tags": {},
      "id": "ocid1.compartment.oc1..00000000000000000000000000000000000000000000000000",
      ...
    },
    ...
  ]
}
```


6. Return to the **Create new policy and compartment** step in the **Add a cloud integration** wizard and paste the **id** value you copied from the response in the last step into the **New compartment-id** field.

7. Click **Next**.

1.4. CREATING A BUCKET FOR ACCESSIBLE COST AND USAGE REPORTS

Create a bucket to store cost and usage reports that that cost management can access.

Procedure

1. In the **Create bucket** step, create a bucket to store cost and usage data so that cost management can access it.

2. Copy the command from the previous step and paste into the Cloud Shell in your Oracle Cloud tab to create a bucket. Refer to the example response for the next steps.

   Example response


```
{
  "data": {
    ...
    "name": "cost-management",
    "namespace": "cost-management-namespace",

    ...
    }
}
```

3. Copy the value pair for the **name** key. In the previous example, this value is **cost-management**.
4. Return to the **Create bucket** step in the **Add a cloud integration** wizard. Paste the value you copied into **New data bucket name**.
5. Return to your Cloud Shell and copy the value for the **namespace** key. In the previous example, copy **cost-management-namespace**.
6. Return to the **Create bucket** step in the **Add a cloud integration** wizard and check your shell prompt for your region. For example, your shell prompt might be **user@cloudshell:~ (uk-london-1)$**. In this example, **uk-london-1** is your region. Copy your region and return to the **Create bucket** step in the **Add a cloud integration** wizard.
7. In the **Create bucket** step in the **Add a cloud integration** wizard, paste your region in **New bucket region**.
8. Click **Next**.

1.5. REPLICATING REPORTS TO A BUCKET

Schedule a task to regularly move the cost information to the bucket you created by creating a function and then a virtual machine to trigger it. In the **Populate bucket** step, visit the link to the script you can use to create a function that must be paired with a virtual machine or CronJob to run daily. The Oracle Cloud documentation provides the following example of how to schedule a recurring job to run the cost transfer script.

NOTE

As non-Red Hat products and documentation can change, instructions for configuring the third-party processes provided in this guide are general and correct at the time of publishing. Contact Oracle Cloud for support.

Procedure

1. In the Oracle Cloud console, open the Navigation menu and click **Developer Services → Functions**.
2. Use the following Python script to create a function application:

```
#
# Copyright 2022 Red Hat Inc.
# SPDX-License-Identifier: Apache-2.0
#
##############################################################################
###############
# Script to collect cost/usage reports from OCI and replicate them to another bucket
#
# Pre-req's you must have a service account or other for this script to gain access to oci
#
# NOTE! You must update the vars below for this script to work correctly
#
# user: ocid of user that has correct permissions for bucket objects
# key_file: Location of auth file for defind user
# fingerprint: Users fingerprint
# tenancy: Tenancy for collecting/copying cost/usage reports
```

```
# region: Home Region of your tenancy
# bucket: Name of Bucket reports will be replicated to
# namespace: Object Storage Namespace
# filename: Name of json file to store last report downloaded default hre is fine
##########################################################################
###############
import datetime
import io
import json
import logging

import oci
from fdk import response


def connect_oci_storage_client(config):
    # Connect to OCI SDK
    try:
        object_storage = oci.object_storage.ObjectStorageClient(config)
        return object_storage
    except (Exception, ValueError) as ex:
        logging.getLogger().info("Error connecting to OCI SDK CLIENT please check
credentials: " + str(ex))


def fetch_reports_file(object_storage, namespace, bucket, filename):
    # Fetch last download report file from bucket
    last_reports_file = None
    try:
        last_reports_file = object_storage.get_object(namespace, bucket, filename)
    except (Exception, ValueError) as ex:
        logging.getLogger().info("Object file does not exist, will attempt to create it: " + str(ex))

    if last_reports_file:
        json_acceptable_string = last_reports_file.data.text.replace("'", "")
        try:
            last_reports = json.loads(json_acceptable_string)
        except (Exception, ValueError) as ex:
            logging.getLogger().info(
                "Json string file not formatted correctly and cannont be parsed, creating fresh file. "
+ str(ex)
            )
            last_reports = {"cost": "", "usage": ""}
    else:
        last_reports = {"cost": "", "usage": ""}

    return last_reports


def get_report_list(object_storage, reporting_namespace, reporting_bucket, prefix, last_file):
    # Create a list of reports
    report_list = object_storage.list_objects(
        reporting_namespace, reporting_bucket, prefix=prefix, start_after=last_file,
fields="timeCreated"
    )
    logging.getLogger().info("Fetching list of cost csv files")
```

```
    return report_list


def copy_reports_to_bucket(
    object_storage,
    report_type,
    report_list,
    bucket,
    namespace,
    region,
    reporting_namespace,
    reporting_bucket,
    last_reports,
):
    # Iterate through cost reports list and copy them to new bucket
    # Start from current month
    start_from = datetime.date.today().replace(day=1)

    if report_list.data.objects != []:
        for report in report_list.data.objects:
            if report.time_created.date() > start_from:
                try:
                    copy_object_details = oci.object_storage.models.CopyObjectDetails(
                        destination_bucket=bucket,
                        destination_namespace=namespace,
                        destination_object_name=report.name,
                        destination_region=region,
                        source_object_name=report.name,
                    )
                    object_storage.copy_object(
                        namespace_name=reporting_namespace,
                        bucket_name=reporting_bucket,
                        copy_object_details=copy_object_details,
                    )
                except (Exception, ValueError) as ex:
                    logging.getLogger().info(f"Failed to copy {report.name} to bucket: {bucket}. " +
str(ex))
        last_reports[report_type] = report.name
    else:
        logging.getLogger().info(f"No new {report_type} reports to copy to bucket: {bucket}.")
    return last_reports


def handler(ctx, data: io.BytesIO = None):
    name = "OCI-cost-mgmt-report-replication-function"
    try:
        body = json.loads(data.getvalue())
        name = body.get("name")
    except (Exception, ValueError) as ex:
        logging.getLogger().info("Error parsing json payload: " + str(ex))

    logging.getLogger().info("Inside Python OCI reporting copy function")

    # PLEASE CHANGE THIS!!!! #
    user = "ocid1.user.oc1..aaaaaa"  # CHANGEME
    key_file = "auth_files/service-account.pem"  # CHANGEME
```

```
    fingerprint = "00.00.00"  # CHANGEME
    tenancy = "ocid1.tenancy.oc1..aaaaaaa"  # CHANGEME
    region = "region"  # CHANGEME
    bucket = "cost-mgmt-bucket"  # CHANGEME
    namespace = "namespace"  # CHANGEME
    filename = "last_reports.json"

    # Get the list of reports
    # https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/clienvironmentvariables.htm!!!
    config = {
        "user": user,
        "key_file": key_file,
        "fingerprint": fingerprint,
        "tenancy": tenancy,
        "region": region,
    }
    # The Object Storage namespace used for OCI reports is bling; the bucket name is the
tenancy OCID.
    reporting_namespace = "bling"
    reporting_bucket = config["tenancy"]
    region = config["region"]

    # Connect to OCI
    object_storage = connect_oci_storage_client(config)

    # Grab reports json and set previously downloaded file values
    last_reports = fetch_reports_file(object_storage, namespace, bucket, filename)
    last_cost_file = last_reports.get("cost")
    last_usage_file = last_reports.get("usage")

    # Get list of cost/usage files
    cost_report_list = get_report_list(
        object_storage, reporting_namespace, reporting_bucket, "reports/cost-csv",
last_cost_file
    )
    usage_report_list = get_report_list(
        object_storage, reporting_namespace, reporting_bucket, "reports/usage-csv",
last_usage_file
    )

    # Copy cost/usage files to new bucket
    last_reports = copy_reports_to_bucket(
        object_storage,
        "cost",
        cost_report_list,
        bucket,
        namespace,
        region,
        reporting_namespace,
        reporting_bucket,
        last_reports,
    )
    last_reports = copy_reports_to_bucket(
        object_storage,
        "usage",
        usage_report_list,
```

```
        bucket,
        namespace,
        region,
        reporting_namespace,
        reporting_bucket,
        last_reports,
    )

    # Save updated filenames to bucket object as string
    object_storage.put_object(namespace, bucket, filename, str(last_reports))

    return response.Response(
        ctx,
        response_data=json.dumps(
            {
                "message": "Last reports saved from {}, Cost: {}, Usage: {}".format(
                    name, last_reports["cost"], last_reports["usage"]
                )
            }
        ),
        headers={"Content-Type": "application/json"},
    )
```

3. Change the values marked **# CHANGEME** to the values for your environment.

```
user = "ocid1.user.oc1..aaaaaa"  # CHANGEME
key_file = "auth_files/service-account.pem"  # CHANGEME
fingerprint = "00.00.00"  # CHANGEME
tenancy = "ocid1.tenancy.oc1..aaaaaaa"  # CHANGEME
region = "region"  # CHANGEME
bucket = "cost-mgmt-bucket"  # CHANGEME
namespace = "namespace"  # CHANGEME
filename = "last_reports.json"
```

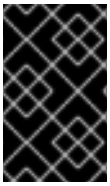
4. Create a virtual machine or a Kubernetes CronJob to trigger your function daily.

1.6. CREATING A BUCKET POLICY TO GRANT READ ACCESS AND FINAL STEPS

Continue in the **Add a cloud integration** wizard by running a command that gives cost management read access to the bucket populated with your Oracle Cloud cost and usage reports.

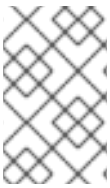
Procedure

1. In the **Populate bucket** step, copy the **oci iam policy create** command and paste into the Cloud Shell in your Oracle Cloud tab to create a read policy.
2. Click **Next**.
3. Review the details of the information you provided. Click **Add**.



IMPORTANT

Oracle Cloud might take several hours to gather and export billing data to cost management. In the meantime, you will receive a **In progress** message, and your integration status will display as **Unknown** in the **Integrations** page.



NOTE

Because third-party products and documentation can change, instructions for configuring the third-party integrations provided are general and correct at the time of publishing. For the most up-to-date information, see the Oracle Cloud documentation.

CHAPTER 2. NEXT STEPS FOR MANAGING YOUR COSTS

After adding your OpenShift Container Platform and Oracle Cloud integrations, in addition to showing cost data by integration, cost management will automatically show Oracle Cloud cost and usage related to running your OpenShift Container Platform clusters on their platform.

On the cost management **Overview** page, your cost data is sorted into **OpenShift** and **Infrastructure** tabs. Select **Perspective** to toggle through different views of your cost data.

You can also use the global navigation menu to view additional details about your costs by cloud provider.

Additional resources

- *Integrating OpenShift Container Platform data into cost management*
- *Integrating Amazon Web Services (AWS) data into cost management*
- *Integrating Google Cloud data into cost management*
- *Integrating Microsoft Azure data into cost management*

2.1. LIMITING ACCESS TO COST MANAGEMENT RESOURCES

After you add and configure integrations in cost management, you can limit access to cost data and resources.

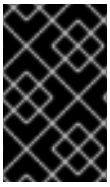
You might not want users to have access to all of your cost data. Instead, you can grant users access only to data that is specific to their projects or organizations. With role-based access control, you can limit the visibility of resources in cost management reports. For example, you can restrict a user's view to only AWS integrations, rather than the entire environment.

To learn how to limit access, see the more in-depth guide *Limiting access to cost management resources* .

2.2. CONFIGURING TAGGING FOR YOUR INTEGRATIONS

The cost management application tracks cloud and infrastructure costs with tags. Tags are also known as labels in OpenShift.

You can refine tags in cost management to filter and attribute resources, organize your resources by cost, and allocate costs to different parts of your cloud infrastructure.



IMPORTANT

You can only configure tags and labels directly on an integration. You can choose the tags that you activate in cost management, however, you cannot edit tags and labels in the cost management application.

To learn more about the following topics, see *Managing cost data using tagging* :

- Planning your tagging strategy to organize your view of cost data
- Understanding how cost management associates tags

- Configuring tags and labels on your integrations

2.3. CONFIGURING COST MODELS TO ACCURATELY REPORT COSTS

Now that you configured your integrations to collect cost and usage data in cost management, you can configure cost models to associate prices to metrics and usage.

A cost model is a framework that uses raw costs and metrics to define calculations for the costs in cost management. You can record, categorize, and distribute the costs that the cost model generates to specific customers, business units, or projects.

In Cost Models, you can complete the following tasks:

- Classifying your costs as infrastructure or supplementary costs
- Capturing monthly costs for OpenShift nodes and clusters
- Applying a markup to account for additional support costs

To learn how to configure a cost model, see *Using cost models*.

2.4. VISUALIZING YOUR COSTS WITH COST EXPLORER

Use cost management Cost Explorer to create custom graphs of time-scaled cost and usage information and ultimately better visualize and interpret your costs.

To learn more about the following topics, see *Visualizing your costs using Cost Explorer*:

- Using Cost Explorer to identify abnormal events
- Understanding how your cost data changes over time
- Creating custom bar charts of your cost and usage data
- Exporting custom cost data tables

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate and prioritize your feedback regarding our documentation. Provide as much detail as possible, so that your request can be quickly addressed.

Prerequisites

- You are logged in to the Red Hat Customer Portal.

Procedure

To provide feedback, perform the following steps:

1. Click the following link: Create Issue.
2. Describe the issue or enhancement in the **Summary** text box.
3. Provide details about the issue or requested enhancement in the **Description** text box.
4. Type your name in the **Reporter** text box.
5. Click the **Create** button.

This action creates a documentation ticket and routes it to the appropriate documentation team. Thank you for taking the time to provide feedback.