Cost Management Service 1-latest

将 Oracle Cloud 数据集成到成本管理

了解如何添加和配置 Oracle Cloud 集成

Last Updated: 2024-10-30

Cost Management Service 1-latest 将 Oracle Cloud 数据集成到成本管理

了解如何添加和配置 Oracle Cloud 集成

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

了解如何将 Oracle Cloud 集成添加到成本管理。成本管理是 Red Hat Insights 服务产品组合的一部分。高级分析工具的 Red Hat Insights 套件可帮助您识别和优先影响您的操作、安全性和业务。

目录

第 1 章 创建 ORACLE 云集成

要将 Oracle Cloud 帐户添加到成本管理中，您必须将 Oracle Cloud 帐户添加为 Red Hat Hybrid Cloud Console 用户界面的集成，并配置 Oracle Cloud 以提供指标。在将 Oracle Cloud 帐户作为数据集成添加到成本管理后，您必须配置功能脚本，将成本和使用量报告复制到成本管理可访问的存储桶。

先决条件

- 具有 云管理员权利的红帽帐户用户
- 访问 Oracle Cloud Console，并可以访问您要添加到成本管理中的比较
- Oracle Cloud 上的服务生成服务用法

在 Oracle Cloud 中完成一些以下步骤，以及 Red Hat Hybrid Cloud Console 中的一些步骤时，请登录两个应用程序并在 Web 浏览器中保持打开状态。首先，使用 **Add a cloud integration**对话框，将 Oracle Cloud 集成添加到 Integrations 页面中的 成本管理。

1.1. 添加 ORACLE CLOUD INFRASTRUCTURE 帐户并命名您的集成

将您的 Oracle Cloud 帐户添加为集成。添加 Oracle Cloud 集成后，成本管理应用程序会处理 Oracle 云帐户中的成本和使用数据，并使其可以被查看。

流程

1. 在 Red Hat Hybrid Cloud Console 中，点 **Settings Menu** > **Integrations**。
2. 在 **Settings** 页面中，单击 **Integrations**。
3. 在 **Cloud** 选项卡中，单击 **Add integration**。
4. 在 **Add a cloud integration**向导中，选择 **Oracle Cloud Infrastructure** 作为集成类型。点击 **Next**。
5. 输入您的集成名称并点 **Next**。
6. 在 **Select application** 步骤中，选择 **Cost management**。点击 **Next**。

1.2. 收集并存储您的全局比较

通过收集全局比较（也称为 Microsoft Azure 中的 **租户 ID**）来继续 **添加云集成** 向导，因此成本管理可以访问您的 Oracle 云比较。

流程

1. **在 Add a cloud integration 向导中，在 Global comparetment-id 步骤中，复制第 1 oci iam 比较列表中的 命令。**
2. **在新标签页中，登录到您的 Oracle Cloud 帐户。**

3. 在菜单栏中，单击 **Developer tools → Cloud Shell**。

4. 在 **Cloud Shell** 窗口中粘贴 **Add a cloud integration** 向导中复制的命令。

5. 在响应中，复制 **compartment-id** 键的值对。在以下示例中，**ID** 从 **ocid1.tenancy.oc1** 开始。

响应示例

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

6. 返回到 **Add a cloud integration** 向导中的 **Global compartment-id** 步骤，并将您的 租户 **ID** 粘贴到 **Global comparetment-id** 字段中。

7. 点击 **Next**。

1.3. 创建用于创建成本和使用报告的策略

通过创建自定义策略和 **Oracle Cloud** 的比较来创建和存储成本和使用报告，继续添加 云集成 向导。

流程

1.

在 Add a cloud integration 向导中，在 Create new policy and comparet ment 页面中，复制 oci iam policy create 命令。

2. 粘贴复制到 Oracle Cloud 选项卡中的 Cloud Shell 的命令，以创建成本和使用报告策略。您还可以添加策略描述。

3. 返回到 Add a cloud integration 向导中的 Create new policy and comparetment 步骤，并复制 oci iam comparetment create 命令。

4. 粘贴复制到 Oracle Cloud 选项卡中的 Cloud Shell 的命令，以创建成本管理比较。

5. 在响应中，复制 id 键的值。在以下示例中，复制包含 ocid1.compartment.oc1 的 id。

响应示例


```
{
 "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. 返回到 Add a cloud integration 向导中的 Create new policy and comparetment 步骤，并将您在上一步中的 id 值粘贴到 New comparetment-id 字段中。

7. 点击 **Next**。

1.4. 为可访问的成本和使用量报告创建存储桶

创建存储桶以存储成本管理可访问的成本和使用情况报告。

流程

1. 在 **Create bucket** 步骤中，创建一个存储桶来存储成本和使用数据，以便成本管理可以访问它。

2. 复制上一步中的命令，并粘贴到 **Oracle Cloud** 选项卡中的 **Cloud Shell** 以创建存储桶。有关后续步骤，请参阅示例响应。

   响应示例

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

    ...
    }
}
```

3. 复制 **name** 键的值对。在上例中，这个值是 **成本管理**。

4. 返回到 **Add a cloud integration** 向导中的 **Create bucket** 步骤。将您复制的值粘贴到 **New data bucket 名称** 中。

5. 返回到 **Cloud Shell**，再复制 **namespace** 键的值。在上例中，复制 **cost-management-namespace**。

6.

返回到 Add a cloud integration 向导中的 Create bucket 步骤，并检查您所在区域的 shell 提示符。例如，您的 shell 提示符可能是 user@cloudshell:~(uk-london-1)$。在本例中，uk-london-1 是您的区域。复制区域，并在 Add a cloud integration 向导中返回到 Create bucket 步骤。

7. 在 Add a cloud integration 向导的 Create bucket 步骤中，将区域粘贴到 New bucket 区域。

8. 点击 Next。

1.5. 将报告复制到存储桶

调度任务，以定期通过创建函数将成本信息移到您创建的存储桶中，然后调度虚拟机来触发它。在 Populate bucket 步骤中，访问脚本的链接，您可以使用它来创建一个必须与虚拟机或 CronJob 配对的功能，以便每天运行。Oracle Cloud 文档提供了如何调度周期性作业以运行 成本传输脚本的示例。

注意

因为非红帽产品和文档可以更改，本指南中提供的第三方流程的说明是常规的，在发布时是正确的。联系 Oracle 云以获取支持。

流程

1. 在 Oracle Cloud 控制台中，打开 Navigation 菜单，再单击 Developer Services → Functions。

2. 使用以下 Python 脚本创建功能应用程序：

```
#
# 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. 将标记为 # CHANGEME 的值改为您的环境的值。

```
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. 创建虚拟机或 Kubernetes CronJob 以每天触发您的功能。

1.6. 创建存储桶策略来授予读取访问权限和最终步骤

通过运行一个命令来继续 Add a cloud integration 向导，它为使用 Oracle Cloud 成本和使用报告填充的存储桶提供成本管理读取访问权限。

流程

1. 在 Populate bucket 步骤中，复制 oci iam policy create 命令，并粘贴到 Oracle Cloud 选项卡中的 Cloud Shell 中，以创建读取策略。

2. 点击 Next。

3. 查看您提供的信息的详细信息。点Add。

重要

Oracle 云可能需要几小时时间来收集并导出计费数据到成本管理。同时，您将收到 In progress 消息，您的集成状态将在 Integrations 页面中显示为 Unknown。

注意

由于第三方产品和文档可能会改变，因此配置第三方集成的说明一般是常规的，并在发布时进行更正。有关最新信息，请查看 Oracle 云文档。

第 2 章 管理成本的后续步骤

添加 OpenShift Container Platform 和 Oracle Cloud 集成后，除了通过集成显示成本数据外，成本管理会自动显示与在其平台上运行 OpenShift Container Platform 集群相关的 Oracle Cloud 成本和使用量。

在 成本管理 概述页上，您的成本数据按照 OpenShift 和 基础架构 选项卡进行排序。选择 Perspective 通过成本数据的不同视图切换。

您还可以使用全局导航菜单查看云供应商成本的更多详情。

其他资源

- *将 OpenShift Container Platform 数据整合到成本管理中*
- *将 Amazon Web Services (AWS)数据整合到成本管理中*
- *将 Google Cloud 数据整合到成本管理中*
- *将 Microsoft Azure 数据整合到成本管理中*

2.1. 限制对成本管理资源的访问

在成本管理中添加和配置集成后，您可以限制对成本数据和资源的访问。

您可能不希望用户访问所有成本数据。相反，您只能向用户授予特定于其项目或机构的数据的访问权限。通过基于角色的访问控制，您可以限制成本管理报告中的资源的可见性。例如，您可以将用户的视图限制为只有 AWS 集成，而不是整个环境。

要了解如何限制访问权限，请参阅更深入的指南 *限制对成本管理资源的访问*。

2.2. 为您的集成配置标记

成本管理应用程序通过标签跟踪云和基础架构成本。在 OpenShift 中，标签也称为标签。

您可以在成本管理中优化标签，过滤和属性资源，按成本组织资源，并为云基础架构的不同部分分配成本。

重要

您只能直接在集成上配置标签和标签。您可以选择在成本管理中激活的标签，但无法在成本管理应用程序中编辑标签和标签。

要了解更多有关以下主题的信息，*请参阅使用标记管理成本数据*：

- 规划标记策略以组织您的成本数据视图
- 了解成本管理关联标签的方式
- 在集成上配置标签和标签

2.3. 配置成本模型以准确报告成本

现在，您已将集成配置为以成本管理方式收集成本和使用数据，您可以配置成本模型，将价格与指标和使用相关联。

成本模型是一个框架，它使用原始成本和指标来定义成本管理成本的计算。您可以记录、分类和分发成本模型给特定客户、业务单元或项目产生的成本。

在 成本 模型中，您可以完成以下任务：

- 将成本分类为基础架构或补充成本
- 捕获 OpenShift 节点和集群的每月成本

- 应用标记以考虑其他支持成本

要了解如何配置成本模型，*请参阅使用成本模型*。

2.4. 使用 COST EXPLORER 可视化您的成本

使用成本管理 Cost Explorer 创建时间扩展成本和使用信息的自定义图形，并最终视觉化并解释您的成本。

要了解有关以下主题的更多信息，*请参阅使用 Cost Explorer 可视化您的成本* ：

- 使用 Cost Explorer 识别异常事件
- 了解如何随着时间推移您的成本数据变化
- 为您的成本和使用数据创建自定义条图表
- 导出自定义成本数据表

对红帽文档提供反馈

我们感谢您对我们文档的反馈并优先排序。尽可能提供更详细的信息，以便可以快速解决您的请求。

先决条件

- 已登陆到红帽客户门户网站。

流程

要提供反馈，请执行以下步骤：

1. 单击以下链接： 创建问题。
2. 描述 Summary 文本框中的问题或增强。
3. 在 Description 文本框中提供有关问题或请求增强的详细信息。
4. 在 Reporter 文本框中输入您的名称。
5. 点 Create 按钮。

此操作会创建一个文档票据，并将其路由到适当的文档团队。感谢您抽出时间提供反馈。