Cost Management Service 1-latest

Oracle Cloud データの Cost Management への統合

Oracle Cloud インテグレーションを追加および設定する方法について

Last Updated: 2024-05-14

Cost Management Service 1-latest Oracle Cloud データの Cost Management への統合

Oracle Cloud インテグレーションを追加および設定する方法について

法律上の通知

概要

Oracle Cloud インテグレーションを Cost Management に追加する方法について説明します。Cost Management は、Red Hat Insights ポートフォリオサービスの一部になります。高度な分析ツールである Red Hat Insights スイートは、運用、セキュリティー、およびビジネスへの影響を特定して優先順位を付けるのに役立ちます。

目次

第1章 ORACLE CLOUD データの COST MANAGEMENT への統合

Cost Management に Oracle Cloud アカウントを追加するには、Red Hat Hybrid Cloud Console ユーザーインターフェイスからインテグレーションとして Oracle Cloud アカウントを追加し、Oracle Cloud を設定して、メトリクスを提供する必要があります。データインテグレーションとして Oracle Cloud アカウントを Cost Management に追加した後、Cost Management がアクセスできるバケットに、コストおよび使用状況レポートをコピーするように関数スクリプトを設定する必要があります。

前提条件

- Cloud Administrator エンタイトルメントを持つ Red Hat アカウントユーザー
- Cost Management に追加するコンパートメントにアクセスできる Oracle Cloud コンソールへのアクセス
- サービスの使用を生成する Oracle Cloud 上のサービス

次の手順の一部を Oracle Cloud で完了し、一部の手順を Red Hat Hybrid Cloud Console で完了する際は、両方のアプリケーションにログインし、Web ブラウザーで開いたままにしておきます。まず、**Add a cloud integration** ダイアログを使用して、Integrations ページから Oracle Cloud インテグレーションを Cost Management に追加します。

1.1. ORACLE CLOUD INFRASTRUCTURE アカウントの追加とインテグレーションの名前付け

Oracle Cloud アカウントをインテグレーションとして追加します。Oracle Cloud インテグレーションを追加した後、Cost Management アプリケーションは Oracle Cloud アカウントからのコストと使用状況データを処理し、表示できるようにします。

手順

1. Red Hat Hybrid Cloud Console から、**Settings Menu** > **Integrations** をクリックします。
2. **Settings** ページで、**Integrations** をクリックします。
3. **Cloud** タブで、**Add integration** をクリックします。
4. **Add a cloud integration** ウィザードで、統合タイプとして **Oracle Cloud Infrastructure** を選択します。**Next** をクリックします。
5. インテグレーションの名前を入力し、**Next** をクリックします。
6. **Select application** の手順で、**Cost management** を選択します。**Next** をクリックします。

1.2. グローバルコンパートメント ID の収集と保存

Add a cloud integration ウィザードを続行し、グローバル **compartment-id** (Microsoft Azure では **tenant-id** とも呼ばれます) を収集して、cost management が Oracle Cloud コンパートメントにアクセスできるようにします。

手順

1. **Add a cloud integration** ウィザードの **Global compartment-id** ステップでステップ 1 **oci iam compartment list** のコマンドをコピーします。

2. 新しいタブで、Oracle Cloud アカウントにログインします。

3. メニューバーで、**Developer tools** > **Cloud Shell** をクリックします。

4. **Add a cloud integration** ウィザードからコピーしたコマンドを **Cloud Shell** ウィンドウに貼り付けます。

5. 応答で、**compartment-id** キーの値のペアをコピーします。以下の例では、ID は **ocid1.tenancy.oc1** で始まります。

   応答の例

```
{
  "data": [
    {
      "compartment-id":
"ocid1.tenancy.oc1..0000000000000000000000000000000000000000000000",
      "defined-tags": {
        "Oracle-Tags": {
            ...
        }
      },
      ...
    }
  ]
}
```

6. **Add a cloud integration** の **Global compartment-id** ステップに戻り、**Global compartment-id** フィールドに **tenant-id** を貼り付けます。

7. **Next** をクリックします。

1.3. コストおよび使用状況レポートを作成するためのポリシーの作成

Oracle Cloud のカスタムポリシーおよびコンパートメントを作成し、コストおよび使用状況レポートを作成および保存して、**Add a cloud integration** ウィザードで続行します。

手順

1. **Add a cloud integration** ウィザードの **Create new policy and compartment**ページで、**oci iam policy create** コマンドをコピーします。

2. コピーしたコマンドを Oracle Cloud タブの Cloud Shell に貼り付けて、コストおよび使用状況レポートのポリシーを作成します。ポリシーの説明を追加することもできます。

3. **Add a cloud integration** ウィザードの **Create new policy and compartment**ステップに戻り、**oci iam compartment create** コマンドをコピーします。

4. コピーしたコマンドを Oracle Cloud タブの Cloud Shell に貼り付けて、Cost Management コンパートメントを作成します。

5. 応答で、**id** キーの値をコピーします。次の例では、**ocid1.compartment.oc1** を含む ID をコピーします。

応答の例

```
{
  "data": [
    {
      "compartment-id": "tenant-id",
      "defined-tags": {
        "Oracle-Tags": {
            ...
        }
      },
      "description": "Cost management compartment for cost and usage data",
      "freeform-tags": {},
      "id": "ocid1.compartment.oc1..0000000000000000000000000000000000000000000000",
      ...
    },
    ...
  ]
}
```

6. **Add a cloud integration** ウィザードの **Create new policy and compartment**ステップに戻り、最後のステップの応答からコピーした **ID** 値を **New compartment-id** フィールドに貼り付けます。

7. **Next** をクリックします。

1.4. アクセス可能なコストおよび使用状況レポートのバケットを作成する

cost management がアクセスできるコストと使用状況レポートを保存するためのバケットを作成します。

手順

1. **Create bucket** ステップで、cost management がアクセスできるように、コストおよび使用状況データを保存するバケットを作成します。

2. 前の手順でコマンドをコピーし、Oracle Cloud タブの Cloud Shell に貼り付けて、バケットを作成します。次の手順については、応答例を参照してください。

応答の例

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

    ...
    }
}
```

3. **name** キーの値のペアをコピーします。前の例では、この値は **cost-management** です。

4. **Add a cloud integration** ウィザードの **Create bucket** ステップに戻ります。コピーした値を **New data bucket name** に貼り付けます。

5. Cloud Shell に戻り、**namespace** キーの値をコピーします。前の例では、**cost-management-namespace** をコピーします。

6. **Add a cloud integration** ウィザードの **Create bucket** ステップに戻り、リージョンのシェルプロンプトを確認します。たとえば、シェルプロンプトは、**user@cloudshell:~ (uk-london-1)$** のようになります。この例では、**uk-london-1** がリージョンです。リージョンをコピーし、**Add a cloud integration** ウィザードの **Create bucket** のステップに戻ります。

7. **Add a cloud integration** ウィザードの **Create bucket** 手順で、**New bucket region** にリージョンを貼り付けます。

8. **Next** をクリックします。

1.5. レポートをバケットに複製する

ファンクションとそれをトリガーする仮想マシンを作成して、コスト情報を作成したバケットに定期的に移動するタスクをスケジュールします。**Populate bucket** ステップで、仮想マシンまたは CronJob と組み合わせて、毎日実行する必要があるファンクションを作成するために使用できるスクリプトへのリンクにアクセスします。Oracle Cloud のドキュメントには、定期的なジョブをスケジュールして、コスト転送スクリプト を実行する方法の次の例が記載されています。

注記

Red Hat 以外の製品およびドキュメントは変更される可能性があるため、このガイドで提供されているサードパーティープロセスの設定手順は一般的であり、公開時点で正しいものです。サポートについては、Oracle Cloud にお問い合わせください。

手順

1. Oracle Cloud コンソール で、ナビゲーションメニューを開いて、**Developer Services** > **Functions** をクリックします。

2. 次の Python スクリプトを使用して、ファンクションアプリケーションを作成します。

```
#
# 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. **# CHANGEME** とマークされた値を環境の値に変更します。

```
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. 仮想マシンまたは Kubernetes CronJob を作成して、ファンクションを毎日トリガーします。

1.6. 読み取りアクセスを許可するバケットポリシーの作成と最終ステップ

Oracle Cloud のコストおよび使用状況レポートが移入されたバケットへの Cost Management 読み取りアクセスを付与するコマンドを実行して、**Add a cloud integration** ウィザードを続行します。

手順

1. **Populate bucket** ステップで、**oci iam policy create** コマンドをコピーし、Oracle Cloud タブの Cloud Shell に貼り付けて、読み取りポリシーを作成します。

2. **Next** をクリックします。

3. 提供した情報の詳細を確認します。**Add** をクリックします。

重要

Oracle Cloud では、請求データを収集して cost management にエクスポートするのに数時間かかる場合があります。その間、**In progress** メッセージが表示され、**Integrations** ページではインテグレーションステータスが **Unknown** と表示されます。

注記

サードパーティー製品およびドキュメントは変更される可能性があるため、提供されるサードパーティー統合を設定するための手順は一般的な内容であり、公開時点では正しいものです。最新情報については、Oracle Cloud のドキュメント を参照してください。

第2章 コストを管理するための次のステップ

OpenShift Container Platform と Oracle Cloud のインテグレーションを追加すると、インテグレーションごとのコストデータに加えて、Cost Management では、プラットフォーム上での OpenShift Container Platform クラスターの実行に関連する Oracle Cloud のコストと使用状況が自動的に表示されます。

cost management **Overview** ページでは、コストデータが **OpenShift** タブと **Infrastructure** タブに分類されます。コストデータのさまざまなビューを切り替えるには、**Perspective** を選択します。

グローバルナビゲーションメニューを使用して、クラウドプロバイダーごとのコストに関する追加の詳細を表示することもできます。

関連情報

- **Cost management への OpenShift Container Platform データの統合**
- **Amazon Web Services (AWS) データの Cost Management への統合**
- **Google Cloud データの Cost Management への統合**
- **Microsoft Azure データの Cost Management への統合**

2.1. COST MANAGEMENT リソースへのアクセス制限

Cost Management でインテグレーションを追加して設定した後、コストデータとリソースへのアクセスを制限できます。

ユーザーがすべてのコストデータにアクセスできる状況は避ける必要がある場合もあります。代わりに、プロジェクトまたは組織に固有のデータにだけアクセスできるようにユーザーにアクセス権を付与できます。ロールベースのアクセス制御を使用すると、Cost Management レポートでのリソースの表示を制限できます。たとえば、ユーザーのビューを環境全体ではなく、AWS インテグレーションのみに制限できます。

アクセスを制限する方法の詳細は、**Cost Management リソースへのアクセス制限** を参照してください。

2.2. インテグレーションのタグ付けの設定

Cost Management アプリケーションは、タグを使用してクラウドとインフラストラクチャーのコストを追跡します。タグは、OpenShift ではラベルとも呼ばれます。

Cost Management でタグを調整して、リソースをフィルタリングおよび属性化し、コスト別にリソースを整理し、クラウドインフラストラクチャーのさまざまな部分にコストを割り当てることができます。

重要

タグとラベルは、インテグレーション上でのみ直接設定できます。Cost Management でアクティブ化するタグの選択はできますが、Cost Management アプリケーションでタグとラベルの編集はできません。

以下のトピックに関する詳細は、**タグ付けを使用したコストデータの管理** を参照してください。

- コストデータの表示を整理するためのタグ付けストラテジーを計画する
- Cost Management がタグを関連付ける方法を理解する
- インテグレーションでタグとラベルを設定する

2.3. コストを正確にレポートするためのコストモデルの設定

Cost Management でコストと使用量のデータを収集するようにインテグレーションを設定したので、価格をメトリクスと使用量に関連付けるコストモデルを設定できます。

コストモデルは、Cost Management において、原価とメトリクスを使用してコスト計算を定義するためのフレームワークです。コストモデルが生成するコストの記録と分類、および特定の顧客、ビジネスユニット、またはプロジェクトに対する配分を行えます。

Cost Models では、次のタスクを完了できます。

- コストを、インフラストラクチャーコストまたは補足コストとして分類します。
- OpenShift ノードおよびクラスターの月額コストを取得します。
- 追加のサポートコストを考慮して利潤を適用します。

コストモデルの設定方法は **コストモデルの使用** を参照してください。

2.4. COST EXPLORER を使用したコストの可視化

Cost Management の Cost Explorer を使用して、時間スケールのコストと使用状況情報のカスタムグラフを作成し、最終的にコストをより適切に可視化して解釈できるようにします。

次のトピックに関する詳細は、**Cost Explorer を使用したコストの可視化** を参照してください。

- Cost Explorer を使用して異常なイベントを特定する。
- 時間の経過とともにコストデータがどのように変化するかを理解する。
- コストおよび使用状況データのカスタムバーチャートを作成する。
- カスタムコストデータテーブルをエクスポートする。

RED HAT ドキュメントへのフィードバック (英語のみ)

エラーを見つけた場合、またはこれらのガイドラインの改善方法に関する案がある場合は、Cost Management Jira ボード で JIRA Issue を作成し、 **ドキュメント** ラベルを追加してください。

フィードバックをお待ちしております。