Hybrid committed spend 1-latest

将 Microsoft Azure 数据集成到混合承诺的花费中

了解如何添加和配置 Microsoft Azure 集成

Last Updated: 2024-06-09

Hybrid committed spend 1-latest 将 Microsoft Azure 数据集成到混合承诺的花费中

了解如何添加和配置 Microsoft Azure 集成

法律通告

摘要

了解如何将 Microsoft Azure 集成添加到混合提交花费。

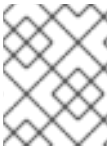
目录

第 1 章 创建 MICROSOFT AZURE 集成

要将 Microsoft Azure 帐户添加到混合提交花费中，您必须将它添加为 Red Hat Hybrid Cloud Console 用户界面的集成，并配置 Microsoft Azure 以提供指标。

要将 Microsoft Azure 帐户配置为混合提交合并，您必须完成以下任务：

1. 创建存储帐户和资源组。
2. 配置存储帐户 Contributor 和 Reader 角色来访问。
3. 创建一个功能来过滤您要发送给红帽的数据。
4. 将每日成本导出调度到红帽可访问的存储帐户。



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，请参阅 Microsoft Azure 文档。

将 Microsoft Azure 集成添加到 Integrations 页面的混合提交成本。

1.1. 添加 MICROSOFT AZURE 帐户并命名您的集成

将 Microsoft Azure 帐户添加为集成，以便混合提交的花费可以处理成本和使用数据。

流程

1. 在 Red Hat Hybrid Cloud Console 中，点 **Settings Menu** > **Integrations**。
2. 在 **Settings** 页面中，在 **Cloud** 选项卡中点 **Add integration**。
3. 点 **Add integration**。
4. 在 **Add a cloud integration** 向导中，选择 **Microsoft Azure** 作为云供应商类型，然后点 **Next**。
5. 输入您的集成名称并点 **Next**。
6. 在 **Select application** 步骤中，选择 **Hybrid committed spend** 并点 **Next**。
7. 在 Specify cost export scope 步骤中，选择 **I am OK，将默认数据发送到成本管理**。
8. 从菜单中选择您的成本数据导出的范围。您可以在订阅级别和其它范围中导出数据。
   a. 为您选择的范围复制生成的命令。
   b. 在 Microsoft Azure 帐户 中，点 **Cloud Shell**。
   c. 粘贴您在上一步中复制的命令。
   d. 从返回的数据复制 **subscription_id** 的值。

   响应示例

```
{
   "subscription_id": 00000000-0000-0000-000000000000
   }
```

9. 在 **Add a cloud integration**向导中，将值粘贴到 **Specify cost export scope** 步骤的 **Cost export scope** 字段中。
10. 点击 **Next**。

1.2. 创建 MICROSOFT AZURE 资源组和存储帐户

在 Microsoft Azure 中创建存储帐户和资源组以存放您的账单导出，以便混合提交的花费可以收集信息。在 **Add a cloud integration**向导 in hybrid committed spend，在 Resource group 和 storage account 页面的字段中输入资源组名称和存储帐户名称。

先决条件

您必须具有具有 Cloud Administrator 权利的红帽用户帐户。

流程

1. 在 Microsoft Azure 帐户 中，搜索 **storage** 并点 **Storage account**。
   a. 在 **Storage accounts** 页面上，单击 **Create**。
   b. 在 **Resource Group** 字段中，单击 **Create new**。输入名称，**然后单击确定**。在本例中，使用 **cost-data-group**。
   c. 在 **Instance details** 部分中，在 **Storage account name**字段中输入名称。例如，使用 **costdata**。
   d. 复制资源组和存储帐户的名称，以便您可以将它们添加到 Red Hat Hybrid Cloud Console 的 **Add a cloud integration**向导中，然后点 **Review**。
   e. 检查存储帐户并点 **Create**。
2. 在 Red Hat Hybrid Cloud Console **Add a cloud integration**向导的 **Resource group and storage account** 页面中，在 **Resource group name** 和 **Storage account name**中输入值。
3. 点击 **Next**。

1.3. 在 MICROSOFT AZURE 中创建每日导出

在 Microsoft Azure 中创建功能，以过滤您的数据并根据常规计划导出数据。exports 创建一个周期性任务，将 Microsoft Azure 成本数据定期发送到存储帐户，该存储帐户存在于资源组中。混合提交的花费必须能够访问资源组以读取 Microsoft Azure 成本数据。这个示例使用 Python 功能来过滤数据并将其发布到之前创建的存储帐户。

流程

1. 要创建导出，请转至 Microsoft Azure 中的 **Portal** 菜单，然后点 **Cost Management + Billing**。
2. 在 Cost Management + Billing 页面上，单击 **Cost Management**。
3. 在 **Settings** 菜单中，在 Cost 管理概览页面中，单击 **Exports**。

4. 要添加导出，请单击 **Add**。
5. 在 **Export details** 部分中，将导出命名为：
6. 在 **Storage** 部分，添加您创建的资源组。

1.4. 查找 MICROSOFT AZURE 订阅 ID

在 Microsoft Azure Cloud Shell 中找到您的 **subscription_id**，并将其添加到 **Add a cloud integration** wizard in hybrid committed spend 中。

流程

1. 在 Microsoft Azure 帐户 中，点 **Cloud Shell**。
2. 输入以下命令获取您的订阅 ID：

```
az account show --query "{subscription_id: id }"
```

3. 从返回的数据复制 **subscription_id** 的值。

   响应示例

```
{
   "subscription_id": 00000000-0000-0000-000000000000
   }
```

4. 将该值粘贴到 **Add a cloud integration**向导的 **Subscription ID** 页面的 Subscription ID 字段中。
5. 点击 **Next**。

1.5. 创建 MICROSOFT AZURE 角色

要授予红帽对 Azure 的访问权限，您必须在 Microsoft Azure 中配置专用凭证。

流程

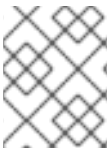
1. 在 **Add a cloud integration**向导的 **Roles** 步骤中，从向导中复制生成的 **az ad sp create-for-rbac** 命令，以创建具有成本管理存储帐户 Contributor 角色的服务主体。
2. 在 Microsoft Azure 帐户 中，点 **Cloud Shell**。
3. 将之前步骤中复制的命令粘贴到云 shell 提示符中。
4. 复制 **租户(Directory) ID**、**Client (Application) ID** 和 **Client secret** 值，并将它们粘贴到 **Add a cloud integration** 向导的 **Roles** 步骤中。
5. 从向导中复制第二个生成的 **az role assignment create** 命令，并将它粘贴到云 shell 提示符中，以创建 Cost Management Reader 角色。
6. 在 **Add a cloud integration**向导中，点 **Next**。
7. 查看向导中提供的信息，然后点 **Add**。

第 2 章 在将 MICROSOFT AZURE 数据集成到混合提交花费前过滤 MICROSOFT AZURE 数据

要使用 RH 共享您的计费数据的子集，您可以在 Microsoft Azure 中配置功能脚本。此脚本复制导出一个对象存储存储桶，然后混合提交的时间可以访问和过滤。

集成 Microsoft Azure 帐户：

1. 创建存储帐户和资源组。
2. 配置存储帐户 Contributor 和 Reader 角色来访问。
3. 创建一个功能来过滤您要发送给红帽的数据。
4. 将每日成本导出调度到红帽可访问的存储帐户。



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，请参阅 Microsoft Azure 文档。

将 Microsoft Azure 集成添加到 Integrations 页面的混合提交成本。

2.1. 添加 MICROSOFT AZURE 帐户并命名您的集成

将 Microsoft Azure 帐户添加为集成，以便混合提交的花费可以处理成本和使用数据。

流程

1. 在 Red Hat Hybrid Cloud Console 中，点 **Settings Menu**  > **Integrations**。
2. 在 **Settings** 页面中，在 **Cloud** 选项卡中点 **Add integration**。
3. 在 **Cloud** 选项卡中，点 **Add integration**。
4. 在 **Add a cloud integration** 向导中，选择 **Microsoft Azure** 作为云供应商类型，然后点 **Next**。
5. 输入您的集成名称并点 **Next**。
6. 在 **Select application** 步骤中，选择 **Hybrid committed spend** 并点 **Next**。
7. 在 Specify cost export scope 步骤中，选择 **I want to manually custom the data set sent to Cost Management** 并点 **Next**。

2.2. 创建 MICROSOFT AZURE 资源组和存储帐户

在 Microsoft Azure 中创建存储帐户，以存放您的账单导出以及第二个存储帐户来存放您的过滤的数据，以便混合提交花费可以收集信息。在 **Add a cloud integration** 向导 in hybrid committed spend，在 Resource group 和 storage account 页面的字段中输入资源组名称和存储帐户名称。

先决条件

您必须具有具有 Cloud Administrator 权利的红帽用户帐户。

流程

1. 在 Microsoft Azure 帐户 中，搜索 **storage** 并点 **Storage account**。
   a. 在 **Storage accounts** 页面上，单击 **Create**。
   b. 在 **Resource Group** 字段中，单击 **Create new**。输入名称，**然后单击确定**。在本例中，使用 **filtered-data-group**。
   c. 在 **Instance details** 部分中，在 **Storage account name**字段中输入名称。例如，使用 **filtereddata**。
   d. 复制资源组和存储帐户的名称，以便您可以将它们添加到 Red Hat Hybrid Cloud Console 的 **Add a cloud integration**向导中，然后点 **Review**。
   e. 检查存储帐户并点 **Create**。
2. 在 Red Hat Hybrid Cloud Console **Add a cloud integration**向导的 **Resource group and storage account** 页面中，在 **Resource group name** 和 **Storage account name**中输入值。
3. 点击 **Next**。

2.3. 查找 MICROSOFT AZURE 订阅 ID

在 Microsoft Azure Cloud Shell 中找到您的 **subscription_id**，并将其添加到 **Add a cloud integration** wizard in hybrid committed spend 中。

流程

1. 在 Microsoft Azure 帐户 中，点 **Cloud Shell**。
2. 输入以下命令获取您的订阅 ID：

```
az account show --query "{subscription_id: id }"
```

3. 从返回的数据复制 **subscription_id** 的值。

   响应示例

```
{
   "subscription_id": 00000000-0000-0000-000000000000
   }
```

4. 将该值粘贴到 **Add a cloud integration**向导的 **Subscription ID** 页面的 Subscription ID 字段中。
5. 点击 **Next**。

2.4. 为您的存储帐户创建 MICROSOFT AZURE 角色

使用 Microsoft Azure Cloud Shell 查找您的租户 **(Directory) ID**、**Client （应用程序）ID** 和 **Client secret**。

流程

1. 在 Microsoft Azure 帐户 中，点 **Cloud Shell**。

2. 输入以下命令获取客户端 ID、secret 和租户名称。将值替换为最后一步中的订阅 ID，将 **resourceGroup1** 替换为之前创建的资源组名称。在本例中，使用 **filtered-data-group**。

```
az ad sp create-for-rbac -n "CostManagement" --role "Storage Account Contributor"  --scope /subscriptions/{subscriptionId}/resourceGroups/{resourceGroup1} --query '{"tenant": tenant, "client_id": appId, "secret": password}'
```

3. 复制 **client_id**、**secret** 和 **租户** 的返回数据中的值。

响应示例

```
{
    "client_id": "00000000-0000-0000-000000000000",
    "secret": "00000000-0000-0000-000000000000",
    "tenant": "00000000-0000-0000-000000000000"
}
```

4. 在 Red Hat Hybrid Cloud Console **中添加云集成** 向导中的 **Role** 步骤中，粘贴 client_id'、**secret** 和 'tenant 的值。

5. 在 Cloud shell 中运行以下命令，以创建成本管理 Reader 角色，并将 **{Client ID}** 替换为上一步中的值。

```
az role assignment create --assignee {Client_ID} --role "Cost Management Reader"
```

6. 点击 **Next**。

2.5. 在 MICROSOFT AZURE 中创建每日导出

在 Microsoft Azure 中创建功能，以过滤您的数据并根据常规计划导出数据。exports 创建一个周期性任务，将 Microsoft Azure 成本数据定期发送到存储帐户，该存储帐户存在于资源组中。混合提交的花费必须能够访问资源组以读取 Microsoft Azure 成本数据。这个示例使用 Python 功能来过滤数据并将其发布到之前创建的存储帐户。

流程

1. 要创建导出，请转至 Microsoft Azure 中的 **Portal** 菜单，然后点 **Cost Management + Billing**。
2. 在 Cost Management + Billing 页面上，单击 **Cost Management**。
3. 在 **Settings** 菜单中，在 Cost 管理概览页面中，单击 **Exports**。
4. 要添加导出，请单击 **Add**。
5. 在 **Export details** 部分中，将导出命名为：
6. 在 **Storage** 部分，添加您创建的资源组。

2.6. 在 MICROSOFT AZURE 中创建功能来过滤您的数据

创建过滤数据的功能，并将其添加到您创建的要与红帽共享的存储帐户中。您可以使用示例 Python 脚本从与红帽费用相关的成本导出中收集成本数据，并将其添加到存储帐户中。

先决条件

- 在您的设备上必须安装 Visual Studio Code。
- 您必须在 Visual Studio Code 中安装 Microsoft Azure 功能扩展。

流程

1. 登录到您的 Microsoft Azure 帐户。
2. 在搜索栏中输入 **functions**，选择 **Functions**，然后单击 **Create**。
3. 为您的功能选择一个托管选项，然后点 **Select**。
4. 在 Create Function App 页面中，通过添加资源组来配置您的功能应用程序。
   a. 在 **Instance Details** 部分中，将功能命名为 app。
   b. 在 **Runtime stack** 中，选择 **Python**
   c. 在 **Version** 中，选择 **3.10**。
5. 点 **Review + create**:
   a. 点 **Create**。
   b. 点 **Go to resource** 来配置函数。
6. 在功能应用程序菜单中，点击 **Functions** 来创建时间触发器功能：
   a. 点 **Create**。
   b. 在开发环境字段中，选择 **VSCode**。
7. 打开 Visual Studio Code，并确保安装了 Microsoft Azure Functions Visual Studio Code 扩展。要创建 Azure 功能，Microsoft 建议使用 Microsoft Visual Studio Code IDE 来开发和部署代码。有关配置 Visual Studio Code 的更多信息，请参阅 Quickstart: 使用 Visual Studio Code 在 Azure 中使用 Python 创建功能 。
   a. 点 Visual Studio Code 中的 Microsoft Azure 选项卡，登录到 Azure。
   b. 在 Visual Studio Code 的 Workspaces 选项卡中，点 **Create function**。
   c. 按照提示为您的功能设置本地位置，并为功能选择语言和版本。在本例中，选择 **Python**，对于 并选择 **Python 3.9**。
   d. 在 **Select a template for your project first function**对话框中，选择 **Timer trigger**，命名功能，然后按 **Enter**
   e. 为希望该函数运行时设置 cron 表达式。在本例中，使用 **0*9*** 在每天上午 9 点运行该功能。
   f. 点 **Create**。
8. 在开发环境中创建功能后，打开 **requirements.txt** 文件，添加以下要求并保存文件：

```
azure-functions
pandas
requests
```

```
azure-identity
azure-storage-blob
```

9. 打开 **__init__.py** 并粘贴以下 Python 脚本。将标记为 **# Required vars 的部分中** 的值更改为您的环境的值。对于 **USER** 和 **PASS** 值，您可以选择使用 Key Vault 凭据 将用户名和密码配置为环境变量。

```
import datetime
import logging
import uuid
import requests
import pandas as pd
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient, ContainerClient

import azure.functions as func


def main(mytimer: func.TimerRequest) -> None:
    utc_timestamp = datetime.datetime.utcnow().replace(
        tzinfo=datetime.timezone.utc).isoformat()

    default_credential = DefaultAzureCredential()

    now = datetime.datetime.now()
    year = now.strftime("%Y")
    month = now.strftime("%m")
    day = now.strftime("%d")
    output_blob_name=f"{year}/{month}/{day}/{uuid.uuid4()}.csv"

    # Required vars to update
    USER = os.getenv('UsernameFromVault')                    # Cost management
username
    PASS = os.getenv('PasswordFromVault')                    # Cost management
password
    integration_id = "<your_integration_id>"                 # Cost management
integration_id
    cost_export_store = "https://<your-cost-export-storage-account>.blob.core.windows.net"
# Cost export storage account url
    cost_export_container = "<your-cost-export-container>"                  # Cost
export container
    filtered_data_store = "https://<your_filtered_data_container-storage-
account>.blob.core.windows.net"        # Filtered data storage account url
    filtered_data_container = "<your_filtered_data_container>"              # Filtered
data container

    # Create the BlobServiceClient object
    blob_service_client = BlobServiceClient(filtered_data_store, credential=default_credential)
    container_client = ContainerClient(cost_export_store, credential=default_credential,
container_name=cost_export_container)

    blob_list = container_client.list_blobs()
    latest_blob = None
    for blob in blob_list:
        if latest_blob:
            if blob.last_modified > latest_blob.last_modified:
```

```
            latest_blob = blob
        else:
            latest_blob = blob

    bc = container_client.get_blob_client(blob=latest_blob)
    data = bc.download_blob()
    blobjct = "/tmp/blob.csv"
    with open(blobjct, "wb") as f:
        data.readinto(f)
    df = pd.read_csv(blobjct)

    filtered_data = df.loc[((df["publisherType"] == "Marketplace") &
((df["publisherName"].astype(str).str.contains("Red Hat")) | (((df["publisherName"] ==
"Microsoft") | (df["publisherName"] == "Azure")) &
(df['meterSubCategory'].astype(str).str.contains("Red Hat") |
df['serviceInfo2'].astype(str).str.contains("Red Hat")))))]

    filtered_data_csv = filtered_data.to_csv (index_label="idx", encoding = "utf-8")

    blob_client = blob_service_client.get_blob_client(container=filtered_data_container,
blob=output_blob_name)

    blob_client.upload_blob(filtered_data_csv, overwrite=True)

    # Post results to console.redhat.com API
    url = "https://console.redhat.com/api/cost-management/v1/ingress/reports/"
    json_data = {"source": integration_id, "reports_list": [f"
{filtered_data_container}/{output_blob_name}"], "bill_year": year, "bill_month": month}
    resp = requests.post(url, json=json_data, auth=(USER, PASS))
    logging.info(f'Post result: {resp}')

    if mytimer.past_due:
        logging.info('The timer is past due!')

    logging.info('Python timer trigger function ran at %s', utc_timestamp)
```

10. 保存该文件。
11. 将功能部署到 Microsoft Azure。

2.7. 配置 MICROSOFT AZURE 角色

配置专用凭证，以授予您的功能 blob 访问权限 Microsoft Azure 成本数据，以便可以将数据从原始存储容器传输到过滤的存储容器。

流程

1. 在 Microsoft Azure 帐户 中，在搜索栏中输入 **功能**。
2. 查找您的功能并选择它。
3. 在 **Settings** 菜单中，单击 **Identity**。
4. 在 Identity 页面上，单击 **Azure 角色分配**。

5. 在 **Role assignments** 页面上，单击 **Add role assignment**。
6. 在 **Scope** 字段中，选择 **Storage** scope。
7. 在 **Resource** 字段中，选择您创建的存储帐户。在本例中，使用 **filtereddata**。
8. 在 role 字段中，选择 **Storage Blob Data Contributor**。
9. 点击 **Save**。
10. 重复这些步骤，为 **Storage Queue Data Contributor** 创建角色。
11. 对您创建的其他存储帐户重复此步骤。在本例中，使用 **billingexportdata**。
12. 在 Red Hat Hybrid Cloud Console 的 **Add a cloud integration** 向导中，点 **Next**。
13. 查看向导中提供的信息，然后点 **Add**。

对红帽文档提供反馈

如果您发现了错误，或者对如何改进这些指南有建议，请在 成本管理 JIRA 板 中创建一个问题并添加 **Documentation** 标签。

非常感谢您的反馈意见！