Cost Management Service 1-latest

Integrating Microsoft Azure data into cost management

Learn how to add and configure your Microsoft Azure integration

Last Updated: 2024-04-24

Cost Management Service 1-latest Integrating Microsoft Azure data into cost management

Learn how to add and configure your Microsoft Azure integration

Legal Notice

Abstract

Learn how to add a Microsoft Azure 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. INTEGRATING MICROSOFT AZURE DATA INTO COST MANAGEMENT

Configure your Microsoft Azure account to allow cost management access.

Configuring your Microsoft Azure account to be a cost management integration requires:

1. Creating a storage account and resource group
2. Choosing the appropriate scope for your cost export
3. Configuring a Storage Account Contributor and Reader roles for access
4. Scheduling daily cost exports

NOTE

As non-Red Hat products and documentation can change without notice, instructions for configuring the third-party integrations provided in this guide are general and correct at the time of publishing. See the Microsoft Azure documentation for the most up-to-date information.

Add your Microsoft Azure integration to cost management from the Integrations page.

1.1. CREATING A MICROSOFT AZURE RESOURCE GROUP AND STORAGE ACCOUNT

Cost export data is written to a storage account, which exists within a resource group. The resource group must be accessible by cost management to read the Microsoft Azure cost data.

Create a new storage account in Microsoft Azure to contain the cost data and metrics that cost management will collect. This requires a resource group; Red Hat recommends creating a dedicated resource group for this storage account.

NOTE

You must have a Red Hat account user with Cloud Administrator entitlements before you can add integrations to cost management.

1. From Red Hat Hybrid Cloud Console , go to the Integrations page and begin adding a Microsoft Azure integration to cost management:
   a. Navigate to **Integrations** and click **Add integration** to open the **Add a cloud integration** wizard.
   b. Enter a name for your integration and click **Next**.
   c. Select **cost management** as the application and **Microsoft Azure** as the integration type. Click **Next**.
2. Create a resource group and storage account in your Microsoft Azure account using the instructions in the See Microsoft Azure documentation *Create a storage account* .

Make a note of the resource group and storage account. They will be needed in subsequent steps.

3. In the Red Hat Hybrid Cloud Console **Add a cloud integration** wizard, enter the **Resource group name** and **Storage account name** and click **Next**.

1.2. CHOOSING A MICROSOFT AZURE COST EXPORT SCOPE

You can create cost export data with differing levels of granularity using **scopes** in Microsoft Azure Cloud. Microsoft Azure supports cost exports scoped as small as a resource group to as large as a billing account containing numerous subscriptions. Depending on the cost data requirements for a given use case, scopes can simplify the configuration of cost management by encompassing multiple subscriptions or isolating access to select resource groups.

NOTE

For more information about how scopes work in Microsoft Azure, see *Understand and work with scopes* in the Azure documentation.

Run one or more of the following commands in the Microsoft Azure Cloud Shell to collect the desired scopes for your cost export:

1. To collect your Resource Group scope, replace *ResourceGroupName* with the name of the Resource Group and run:

```
$ az group show --name {ResourceGroupName} | jq .id | tr -d ""
```

2. To collect your Subscription scope, run:

```
$ az account show --query "{ id: id }" | jq '.id' | tr -d "" | awk '{print "/subscriptions/"$0}'
```

3. To collect your Billing Account scope, replace *billingAccountName* with the name of the Billing Account and run:

```
$ az billing account show --name "{billingAccountName}" | jq '.id' | tr -d ""
```

4. To collect your Enrollment Account scope, replace *enrollmentAccountName* with the name of the Enrollment Account and run:

```
$ az billing enrollment-account show --name "{enrollmentAccountName}" | jq '.id' | tr -d ""
```

5. To collect your Management Group scope, replace *GroupName* with the name of the Management Group and run:

```
$ az account management-group show --name "{GroupName}" | jq '.id' | tr -d ""
```

6. To collect your Billing Profile scope, replace *billingAccountName* and *billingProfileName* with the names of the Billing Account and Billing Profile and run:

```
$ az billing profile show --account-name "{billingAccountName}" --name "
{billingProfileName}" | jq '.id' | tr -d ""
```

7. To collect your Invoice Section scope, replace *billingAccountName*, *billingProfileName*, *invoiceSectionName* with the names of the Billing Account, Billing Profile, and Invoice Section and run:

```
$ az billing invoice section show --account-name "{billingAccountName}" --profile-name "
{billingProfileName}" --name "{invoiceSectionName}" | jq '.id' | tr -d ""
```

1.3. CONFIGURING MICROSOFT AZURE ROLES

Red Hat recommends configuring dedicated credentials to grant cost management read-only access to Microsoft Azure cost data. Configure a service principal with Storage Account Contributor and Reader role in Azure to provide this access to cost management.

1. In Microsoft Azure Cloud Shell, run the following command to obtain your Subscription ID:

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

2. In the Red Hat Hybrid Cloud Console **Add a cloud integration** wizard, enter your **Subscription ID**. Click **Next** to move to the next screen.

3. In Microsoft Azure Cloud Shell, run the following command to create a cost management Storage Account Contributor role, and obtain your tenant ID, client (application) ID, and client secret:

```
$ 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}'
```

4. In the Red Hat Hybrid Cloud Console **Add a cloud integration** wizard, enter your Microsoft Azure **Tenant ID**, **Client ID**, and **Client Secret**.

5. Create a Reader role in Microsoft Azure for cost management.

   a. If the cost export scope is for an Enterprise Agreement (EA) account then launch the Microsoft Azure Enterprise Portal to give the service principal created previously an Administrator role on the account. For more information, see Assign access to Cost Management data in the Azure documentation.

   b. If the cost export scope is for a billing account, billing profile, or invoice section in a Microsoft Customer Agreement (MCA), launch the Cost Management and Billing service in the Microsoft Azure portal. Select the appropriate scope and give the service principal created above the appropriate Reader role from the IAM view. For more information, see Understand Microsoft Customer Agreement administrative roles in Microsoft Azure in the Azure documentation.

   c. If the cost export scope is for a resource group, subscription, or management group then in Microsoft Azure Cloud Shell, run the following command to create a cost management Reader role:

```
$ az role assignment create --assignee "<your_Client_ID>" --role "Cost Management
Reader" --scope {costExportScope}
```

6. Click **Next**.

1.4. CONFIGURING A DAILY MICROSOFT AZURE DATA EXPORT SCHEDULE

Create a recurring task to export your cost data on a daily basis automatically to your Microsoft Azure storage account, where cost management will retrieve the data at the desired scope.

1. In Microsoft Azure, add a new export as described in the instructions in the Azure article *Create and manage exported data*.
    - Select a **Name** for the export that should be supplied in the Red Hat Hybrid Cloud Console **Add a cloud integration** wizard.
    - For **Export type**, select **Daily export of month-to-date costs**.
    - For **Storage account**, select the account you created earlier.
    - Enter any value for the container name and directory path for the export. These values provide the tree structure in the storage account where report files are stored.
    - Click **Run now** to start exporting data to the Microsoft Azure storage container.
2. In the Red Hat Hybrid Cloud Console **Add a cloud integration** wizard, click **Next** when you have created the export schedule and review the integration details.
3. Click **Finish** to complete adding the Microsoft Azure integration to cost management.

After the schedule is created, cost management will begin polling Microsoft Azure for cost data, which will appear on the cost management dashboard.

CHAPTER 2. FILTERING YOUR MICROSOFT AZURE DATA BEFORE INTEGRATING IT INTO COST MANAGEMENT

You can configure a function script in Microsoft Azure to copy the cost exports and object storage bucket that cost management can access and filter your data to share a subset of your billing data with Red Hat.

To configure your Microsoft Azure account to be a cost management integration:

1. Create a storage account and resource group.
2. Configure Storage Account Contributor and Reader roles for access.
3. Create a function to filter the data you want to send to Red Hat.
4. Schedule daily cost exports to a storage account accessible to Red Hat.

NOTE

Because third-party products and documentation that are not part of Red Hat can change without notice, 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 Microsoft Azure documentation.

Add your Microsoft Azure integration to cost management from the Integrations page.

2.1. ADDING A MICROSOFT AZURE ACCOUNT AND NAMING YOUR INTEGRATION

Add your Microsoft Azure account as an integration so the cost management application can process the cost and usage data.

Procedure

1. From Red Hat Hybrid Cloud Console , click **Settings Menu** > (**Settings**).
2. On the **Settings** page, click **Integrations**.
3. In the **Cloud** tab, click **Add integration**.
4. In the **Add a cloud integration**wizard, select **Microsoft Azure** as the cloud provider type and click **Next**.
5. Enter a name for your integration and click **Next**.
6. In the **Select application** step, select **cost management** and click **Next**.
7. In the Specify cost export scope step, select **I wish to manually customize the data set sent to Cost Management** and click **Next**.

2.2. CREATING A MICROSOFT AZURE RESOURCE GROUP AND STORAGE ACCOUNT

Create a storage account in Microsoft Azure to house the cost data and metrics and a storage account to house your filtered cost data that cost management will collect. After you create the resource group and storage account, you can paste the resource group name and storage account name in the fields in the Resource group and storage account page in the **Add a cloud integration** wizard in cost management.

Prerequisites

- You must have a Red Hat user account with Cloud Administrator entitlements.

Procedure

1. In your Microsoft Azure account, search for **storage** and click **Storage accounts**.
   a. On the **Storage accounts** page, click **Create**.
   b. On the **Create a storage account** page, in the **Resource Group** field, click **Create new**. Enter a name, and click **OK**. In this example, use **filtered-data-group**.
   c. In the instance details section, enter a name in the **Storage account name** field. In this example, use **filtereddata**.
   d. Make a note of the name of the resource group and storage account so you can add them to the **Add a cloud integration** wizard in Red Hat Hybrid Cloud Console and click **Review**.
   e. Review the storage account and click **Create**.
2. In the Red Hat Hybrid Cloud Console **Add a cloud integration** wizard, on the **Resource group and storage account** page, enter values in the **Resource group name** and **Storage account name**.
3. Click **Next**.

2.3. FINDING YOUR MICROSOFT AZURE SUBSCRIPTION ID

Use the Microsoft Azure Cloud Shell to find your **subscription_id** and add it to the **Add a cloud integration** wizard in cost management.

Procedure

1. In your Microsoft Azure account, click **Cloud Shell**.
2. Enter the following command to obtain your Subscription ID:

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

3. Copy the value for the **subscription_id** from the returned data.

Example response

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

4. Paste that value in the **Subscription ID** field on the Subscription ID page in the **Add a cloud integration** wizard in Red Hat Hybrid Cloud Console.

5. Click **Next**.

2.4. CREATING MICROSOFT AZURE ROLES FOR YOUR STORAGE ACCOUNT

Use the Microsoft Azure Cloud Shell to find your **Tenant (Directory) ID**, **Client (Application) ID**, and **Client secret**.

Procedure

1. In your Microsoft Azure account, click **Cloud Shell**.

2. Enter the following command to get your client ID, secret, and tenant name. Replace the values with your subscription ID from the last step and **resourceGroup1** with the resource group name you created before. In this example, use **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. Copy the values from the returned data for the **client_id**, **secret**, and **tenant**.

   Example response

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

4. Paste the values of client_id`, **secret**, and `tenant in the **Roles** step in the **Add a cloud integration** wizard in Red Hat Hybrid Cloud Console.

5. Run the following command in the Cloud shell to create a Cost Management Reader role and replace **{Client ID}** with the value from the previous step.

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

6. Click **Next**.

2.5. CREATING A DAILY EXPORT IN MICROSOFT AZURE

Create a function in Microsoft Azure to filter your data and export it on a regular schedule. Exports create a recurring task that sends your Microsoft Azure cost data regularly to a storage account, which exists within a resource group. Cost management must be able to access the resource group to read the Microsoft Azure cost data. This example uses a Python function to filter the data and post it to the storage account you created earlier.

Procedure

1. To create the export, go to the **Portal** menu in Microsoft Azure and click **Cost Management + Billing**.
2. On the Cost Management + Billing page, click **Cost Management**.
3. In the **Settings** menu, in the Cost management overview page, click, **Exports**.
4. To add an export, click **Add**.
5. In the **Export details** section, name the export.
6. In the **Storage** section, add the resource group you created.

2.6. CREATING A FUNCTION IN MICROSOFT AZURE TO FILTER YOUR DATA

Create the function that filters your data and adds it to the storage account that you created to share with Red Hat. You can use the example Python script to gather the cost data from your cost exports related to your Red Hat expenses and add it to the storage account.

Prerequisites

- You must have Visual Studio Code installed on your device.
- You must have the Microsoft Azure functions extension installed in Visual Studio Code.

Procedure

1. Log in to your Microsoft Azure account. To begin creating the function app, type **functions** in the search bar, select **Functions**, and click **Create**.
    a. On the Create Function App page, configure your function app by adding your resource group.
    b. In the **Instance Details** section, name your function app.
    c. For runtime stack, select **Python**
    d. For version, select **3.10**.
2. Click **Review + create**:
    a. Click **Create**.
    b. Click **Go to resource** to configure the function.
3. In the function app menu, click **Functions** to create a time trigger function:
    a. Click **Create**.
    b. In the development environment field, select **VSCode**.
4. Open Visual Studio Code and ensure that the Microsoft Azure Functions Visual Studio Code extension is installed. To create an Azure function, Microsoft recommends that you use their Microsoft Visual Studio Code IDE to develop and deploy code. For more information about configuring Visual Studio Code, see Quickstart: Create a function in Azure with Python using Visual Studio Code .

a. Click the Microsoft Azure tab in Visual Studio Code, sign in to Azure.

b. In the workspaces tab in Visual Studio Code, click **Create function**.

c. Follow the prompts to set a local location for your function and select a language and version for your function. In this example, select **Python**, and select **Python 3.9**.

d. In the **Select a template for your project's first function** dialog, select **Timer trigger**, name the function, and press **Enter**

e. Set the cron expression for when you want the function to run. In this example, use **0*9*** to run the function daily at 9 AM.

f. Click **Create**.

5. After you create the function in your development environment, open the **requirements.txt** file, add the following requirements, and save the file:

```
azure-functions
pandas
requests
azure-identity
azure-storage-blob
```

6. Open **__init__.py** and paste the following Python script. Change the values in the section marked **# Required vars to update** to the values for your environment. For the **USER** and **PASS** values, you can optionally use Key Vault Credentials to configure your username and password as environment variables.

```
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)
```

7. Save the file.
8. Deploy the function to Microsoft Azure.

2.7. CONFIGURING MICROSOFT AZURE ROLES

Configure dedicated credentials to grant your function blob access to Microsoft Azure cost data so it can transfer the data from the original storage container to the filtered storage container.

Procedure

1. In your Microsoft Azure account, type **functions** in the search bar.
2. Find your function and select it.
3. In the **Settings** menu, click **Identity**.
4. On the Identity page, click **Azure role assignments**.
5. On the **Role assignments** page, click **Add role assignment**
6. In the **Scope** field, select the **Storage** scope.
7. In the **Resource** field, select the storage account that you created. In this example, use **filtereddata**.
8. In the role field, select **Storage Blob Data Contributor**.
9. Click **Save**.
10. Repeat these steps to create a role for **Storage Queue Data Contributor**.
11. Repeat this process for the other storage account that you created. In this example, use **billingexportdata**.
12. In the **Add a cloud integration**wizard in Red Hat Hybrid Cloud Console , click **Next**.
13. Review the information you provided in the wizard and click **Add**.

CHAPTER 3. NEXT STEPS FOR MANAGING YOUR COSTS

After adding your OpenShift Container Platform and Microsoft Azure integration, in addition to showing cost data by integration, cost management will automatically show Azure 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 Oracle Cloud data into cost management*

3.1. LIMITING ACCESS TO COST MANAGEMENT RESOURCES

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

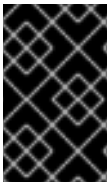
You might not want users to have access to all of your cost data. Instead, you can grant users access only to data that is specific to their projects or organizations. With role-based access control, you can limit the visibility of resources in cost management reports. For example, you can restrict a 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* .

3.2. 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

3.3. 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*.

3.4. 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

If you found an error or have a suggestion on how to improve these guidelines, open an issue in the cost management Jira board and add the **Documentation** label.

We appreciate your feedback!