Hybrid committed spend 1-latest

Integrating Microsoft Azure data into hybrid committed spend

Learn how to add and configure your Microsoft Azure integrations

Last Updated: 2024-06-07

Hybrid committed spend 1-latest Integrating Microsoft Azure data into hybrid committed spend

Learn how to add and configure your Microsoft Azure integrations

Abstract

Learn how to add a Microsoft Azure integration to hybrid committed spend.

Table of Contents

CHAPTER 1. CREATING A MICROSOFT AZURE INTEGRATION **3**

1.1. ADDING A MICROSOFT AZURE ACCOUNT AND NAMING YOUR INTEGRATION 3

1.2. CREATING A MICROSOFT AZURE RESOURCE GROUP AND STORAGE ACCOUNT 4

1.3. CREATING A DAILY EXPORT IN MICROSOFT AZURE 4

1.4. FINDING YOUR MICROSOFT AZURE SUBSCRIPTION ID 5

1.5. CREATING MICROSOFT AZURE ROLES 5

CHAPTER 2. FILTERING YOUR MICROSOFT AZURE DATA BEFORE INTEGRATING IT INTO HYBRID COMMITTED SPEND **7**

2.1. ADDING A MICROSOFT AZURE ACCOUNT AND NAMING YOUR INTEGRATION 7

2.2. CREATING A MICROSOFT AZURE RESOURCE GROUP AND STORAGE ACCOUNT 7

2.3. FINDING YOUR MICROSOFT AZURE SUBSCRIPTION ID 8

2.4. CREATING MICROSOFT AZURE ROLES FOR YOUR STORAGE ACCOUNT 9

2.5. CREATING A DAILY EXPORT IN MICROSOFT AZURE 9

2.6. CREATING A FUNCTION IN MICROSOFT AZURE TO FILTER YOUR DATA 10

2.7. CONFIGURING MICROSOFT AZURE ROLES 13

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION **14**

CHAPTER 1. CREATING A MICROSOFT AZURE INTEGRATION

To add an Microsoft Azure account to hybrid committed spend, you must add it as a integration from the Red Hat Hybrid Cloud Console user interface and configure Microsoft Azure to provide metrics.

To configure your Microsoft Azure account to be an hybrid committed spend integration, you must complete the following tasks:

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 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 Microsoft Azure documentation.

Add your Microsoft Azure integration to hybrid committed spend from the Integrations page.

1.1. ADDING A MICROSOFT AZURE ACCOUNT AND NAMING YOUR INTEGRATION

Add your Microsoft Azure account as an integration so hybrid committed spend can process the cost and usage data.

Procedure

1. From Red Hat Hybrid Cloud Console, click **Settings Menu** > **Integrations**.
2. On the **Settings** page, in the **Cloud** tab, click **Add integration**.
3. 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 **Hybrid committed spend** and click **Next**.
7. In the Specify cost export scope step, select **I am OK with sending the default data to Cost Management**.
8. Select the scope of your cost data export from the menu. You can export data at the subscription level and other scopes in your subscription.
    a. Copy the generated command for the scope you selected.
    b. In your Microsoft Azure account, click **Cloud Shell**.

c. Paste the command you copied from the earlier step.

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

Example response

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

9. In the **Add a cloud integration** wizard, paste the value into the **Cost export scope** field on the **Specify cost export scope** step.

10. Click **Next**.

1.2. CREATING A MICROSOFT AZURE RESOURCE GROUP AND STORAGE ACCOUNT

Create a storage account and resource group in Microsoft Azure to house your billing exports so that hybrid committed spend can collect the information. In the **Add a cloud integration** wizard in hybrid committed spend, enter the resource group name and storage account name in the fields in the Resource group and storage account page.

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. In the **Resource Group** field, click **Create new**. Enter a name, and click **OK**. In this example, use **cost-data-group**.
   c. In the **Instance details** section, enter a name in the **Storage account name** field. For example, use **costdata**.
   d. Copy the names 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**.

1.3. 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. Hybrid committed spend must be able to access the resource group toread 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.

1.4. FINDING YOUR MICROSOFT AZURE SUBSCRIPTION ID

Find your **subscription_id** in the Microsoft Azure Cloud Shell and add it to the **Add a cloud integration** wizard in hybrid committed spend.

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.
5. Click **Next**.

1.5. CREATING MICROSOFT AZURE ROLES

To grant Red Hat access to Azure, you must configure dedicated credentials in Microsoft Azure.

Procedure

1. In the **Add a cloud integration** wizard, on the **Roles** step, copy the generated **az ad sp create-for-rbac** command from the wizard to create a service principal with the Cost Management Storage Account Contributor role.

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

3. Paste the command you copied in the earlier step in the cloud shell prompt.

4. Copy the **Tenant (Directory) ID**, **Client (Application) ID**, and **Client secret** values and paste them into the **Roles** step of the **Add a cloud integration** wizard.

5. Copy the second generated **az role assignment create** command from the wizard and paste it in the cloud shell prompt to create a Cost Management Reader role.

6. In the **Add a cloud integration** wizard, click **Next**.

7. Review the information you provided in the wizard and click **Add**.

CHAPTER 2. FILTERING YOUR MICROSOFT AZURE DATA BEFORE INTEGRATING IT INTO HYBRID COMMITTED SPEND

To share a subset of your billing data with RH, you can configure a function script in Microsoft Azure. This script copies exports an object storage bucket that hybrid committed spend can then access and filter.

To integrate your Microsoft Azure account:

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 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 Microsoft Azure documentation.

Add your Microsoft Azure integration to hybrid committed spend from the Integrations page.

2.1. ADDING A MICROSOFT AZURE ACCOUNT AND NAMING YOUR INTEGRATION

Add your Microsoft Azure account as an integration so hybrid committed spend can process the cost and usage data.

Procedure

1. From Red Hat Hybrid Cloud Console , click **Settings Menu** ⚙ > **Integrations**.
2. On the **Settings** page, in the **Cloud** tab, click **Add integration**.
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 **Hybrid committed spend** 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 your billing exports and a second storage account to house your filtered data so that hybrid committed spend can collect the information. In the **Add a cloud integration** wizard in hybrid committed spend, enter the resource group name and storage account name in the fields in the Resource group and storage account page.

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. 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. For example, use **filtereddata**.
    d. Copy the names 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

Find your **subscription_id** in the Microsoft Azure Cloud Shell and add it to the **Add a cloud integration** wizard in hybrid committed spend.

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.

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. Hybrid committed spend must be able to access the resource group toread 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.
2. Enter **functions** in the search bar, select **Functions**, and click **Create**.
3. Select a hosting option for your function and click **Select**.
4. On the Create Function App page, configure your function app by adding your resource group. =
   a. In the **Instance Details** section, name your function app.
   b. In **Runtime stack**, select **Python**
   c. In **Version**, select **3.10**.
5. Click **Review + create**:
   a. Click **Create**.
   b. Click **Go to resource** to configure the function.
6. In the function app menu, click **Functions** to create a time trigger function:
   a. Click **Create**.
   b. In the development environment field, select **VSCode**.
7. 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**, for 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**.

8. 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
```

9. 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)
```

10. Save the file.

11. 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**.

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!