Introduction

This tutorial introduces you to account-level credit usage monitoring with Budgets by setting up the account budget and creating a custom budget that monitors a group of specified objects.

With budgets, you can monitor credit usage for the compute costs of supported objects, including credit usage for background maintenance tasks and serverless features. Budgets enables you to set a monthly spending limit for each budget and sends a notification email when your current spending is projected to exceed the monthly spending limit.

You can complete this tutorial using a worksheet in Snowsight or using a CLI client such as SnowSQL. Some portions of this tutorial can be completed using Snowsight.

By the end of this tutorial, you will learn how to do the following:

- Create custom roles to monitor and manage budgets.
- Grant the required privileges to add objects to a custom budget.
- Activate and set up an account budget.
- Create a custom budget and add objects to it.

Prerequisites¶

To complete this tutorial, the following prerequisites are required:

- You must be able to use the ACCOUNTADMIN role to create the roles used in this tutorial.
- You must verify your email address. Only verified email addresses can be added to a budget notification list.

Create a notification integration

Budgets use a notification integration to send notification emails when current credit usage is expected to exceed the monthly spending limit. The `ALLOWED_RECIPIENTS` list *must* include the verified email addresses of the users to receive budgets notifications.

A notification integration is required if you are completing the tutorial using SQL. Follow the steps below to create one.

When you use Snowsight to set up a budget, the notification integration is automatically created for you. If you are going to use Snowsight to set up your budgets, you can skip to the next step.

Execute the following statement to create a notification integration. Use your verified email address in the ALLOWED_RECIPIENTS list:

```
USE ROLE ACCOUNTADMIN;

CREATE NOTIFICATION INTEGRATION budgets_notification_integration
  TYPE=EMAIL
  ENABLED=TRUE
  ALLOWED_RECIPIENTS=('<YOUR_EMAIL_ADDRESS>');
```

After you create the notification integration, grant the USAGE privilege to the SNOWFLAKE application. This privilege is required in order for Budgets to use the notification integration to send emails.
Execute the following statement to grant the USAGE privilege on the notification integration:

```
GRANT USAGE ON INTEGRATION budgets_notification_integration
  TO APPLICATION snowflake;
```

Create a database, schema, and custom roles

In this step, the following objects are created for the tutorial to create, manage, and monitor budgets:

- A database and schema in which to create custom budgets.
- A custom role to manage the account budget.
- A custom role to monitor the account budget.
- A custom role to create custom budgets.

Create a database and schema in which to create a custom budget using the following steps:

Create the database and schema in which to create the custom budget:

```
USE ROLE ACCOUNTADMIN;

CREATE OR REPLACE DATABASE budgets_db;

CREATE OR REPLACE  SCHEMA budgets_db.budgets_schema;
```

Create custom role `account_budget_admin` for the account budget administrator. The account budget administrator can take the following actions on the account budget:

a. Activate and deactivate the account budget.
b. Set the spending limit.
c. Edit notification settings.
d. Monitor credit usage for the account.

```
USE ROLE ACCOUNTADMIN;

CREATE ROLE account_budget_admin;

GRANT APPLICATION ROLE SNOWFLAKE.BUDGET_ADMIN TO ROLE account_budget_admin;

GRANT IMPORTED PRIVILEGES ON DATABASE SNOWFLAKE TO ROLE account_budget_admin;
```

Create custom role `account_budget_monitor` to be granted to account budget monitors. An account budget monitor can take the following actions on the account budget:

e. Monitor credit usage for the account.
f. View the email notification settings.
g. View the monthly spending limit for the account.

```
USE ROLE ACCOUNTADMIN;

CREATE ROLE account_budget_monitor;

GRANT APPLICATION ROLE SNOWFLAKE.BUDGET_VIEWER TO ROLE account_budget_monitor;

GRANT IMPORTED PRIVILEGES ON DATABASE SNOWFLAKE TO ROLE account_budget_monitor;
```

Create a custom role `budget_owner` with the required role and privileges to create custom budgets in the schema `budgets_db.budgets_schema`:

```
USE ROLE ACCOUNTADMIN;

CREATE ROLE budget_owner;

GRANT USAGE ON DATABASE budgets_db TO ROLE budget_owner;
GRANT USAGE ON SCHEMA budgets_db.budgets_schema TO ROLE budget_owner;

GRANT DATABASE ROLE SNOWFLAKE.BUDGET_CREATOR TO ROLE budget_owner;

GRANT CREATE SNOWFLAKE.CORE.BUDGET ON SCHEMA budgets_db.budgets_schema
  TO ROLE budget_owner;
```

Create two custom roles to manage and monitor custom budgets. These roles will be granted additional privileges later in the tutorial after the custom budget is created. To create the custom roles, follow these steps:

Create a custom `budget_admin` role that can manage and monitor a custom budget:

```
USE ROLE ACCOUNTADMIN;

CREATE ROLE budget_admin;

GRANT USAGE ON DATABASE budgets_db TO ROLE budget_admin;

GRANT USAGE ON SCHEMA budgets_db.budgets_schema TO ROLE budget_admin;

GRANT DATABASE ROLE SNOWFLAKE.USAGE_VIEWER TO ROLE budget_admin;

CREATE ROLE budget_monitor;

GRANT USAGE ON DATABASE budgets_db TO ROLE budget_monitor;

GRANT USAGE ON SCHEMA budgets_db.budgets_schema TO ROLE budget_monitor;

GRANT DATABASE ROLE SNOWFLAKE.USAGE_VIEWER TO ROLE budget_monitor;
```

Grant custom budget roles to yourself to use in future steps of the tutorial:

Grant the roles to yourself:

```
GRANT ROLE account_budget_admin
  TO USER <YOUR_USER_NAME>;
GRANT ROLE account_budget_monitor
  TO USER <YOUR_USER_NAME>;
GRANT ROLE budget_owner
  TO USER <YOUR_USER_NAME>;
GRANT ROLE budget_monitor
  TO USER <YOUR_USER_NAME>;
```

Create the objects for the custom budget

In this step, create objects to add to a custom budget and grant privileges to the custom roles you created in the previous step. You will be creating the following objects:

- A warehouse to add to a custom budget.
- A database to add to a custom budget.

Create a warehouse and grant the USAGE and APPLYBUDGET privileges on the warehouse to the custom roles you created. The APPLYBUDGET privilege is required to add an object to a budget.

- Create warehouse `na_finance_wh`:
- Grant the USAGE privilege to custom budget roles:
- Grant the APPLYBUDGET privilege on the warehouse to role `budget_owner`:

```
CREATE WAREHOUSE na_finance_wh;
GRANT USAGE ON WAREHOUSE na_finance_wh TO ROLE account_budget_admin;
GRANT USAGE ON WAREHOUSE na_finance_wh TO ROLE account_budget_monitor;
GRANT USAGE ON WAREHOUSE na_finance_wh TO ROLE budget_admin;
GRANT USAGE ON WAREHOUSE na_finance_wh TO ROLE budget_owner;
GRANT USAGE ON WAREHOUSE na_finance_wh TO ROLE budget_monitor;
GRANT APPLYBUDGET ON WAREHOUSE na_finance_wh TO ROLE budget_owner;
```

Create a database and grant the APPLYBUDGET privilege on the warehouse to the custom budget owner role you created. The APPLYBUDGET privilege is required to add an object to a budget.

```
CREATE DATABASE na_finance_db;
GRANT APPLYBUDGET ON DATABASE  na_finance_db TO ROLE budget_owner;
```

Activate and set up the account budget

The account budget monitors credit usage for the compute costs of all Budgets supported objects in the account, including background maintenance tasks (for example, automatic clustering) and serverless features. The account budget must be activated before it can start monitoring credit usage. After it is activated, you can set the monthly spending limit for the account and the email list of notification recipients. Budgets sends a notification email when current credit usage is expected to exceed the monthly spending limit.

Activate and set up the account budget using the following steps:

- Use the `account_budget_admin` role you created in a previous step to activate the account budget
- Set the spending limit for the account budget to 500 credits per month:

```
USE ROLE account_budget_admin;

CALL snowflake.local.account_root_budget!ACTIVATE();
CALL snowflake.local.account_root_budget!SET_SPENDING_LIMIT(500);
```

To set up the email notification list, use your verified email address and the notification integration you created earlier in the tutorial:

```
CALL snowflake.local.account_root_budget!SET_EMAIL_NOTIFICATIONS(
   'budgets_notification_integration',
   '<YOUR_EMAIL_ADDRESS>');
```

Create a custom budget

Now that you have activated and set up your account budget, create a custom budget to monitor the credit usage in your account for a specified group of objects. For this tutorial, add the `na_finance_wh` warehouse and `na_finance_db` to the custom budget.

Create custom budgets with the following steps:

Use the `budget_owner` role to create budget `na_finance_budget` in `budgets_db.budgets_schema`:

```
-- Lab 8.5
USE ROLE budget_owner;
USE SCHEMA budgets_db.budgets_schema;
USE WAREHOUSE na_finance_wh;

CREATE SNOWFLAKE.CORE.BUDGET na_finance_budget();
```

Set the monthly spending limit and email notification list for budget `na_finance_budget` using the following steps:

- Set the monthly spending limit to 500 credits:
- To set up the notification list, use your verified email address and the notification integration created in the first step of the tutorial:
- Add database na_finance_db and warehouse na_finance_wh to budget na_finance_budget

```
CALL na_finance_budget!SET_SPENDING_LIMIT(500);
CALL na_finance_budget!SET_EMAIL_NOTIFICATIONS('budgets_notification_integration',
                                               '<YOUR_EMAIL_ADDRESS>');
CALL na_finance_budget!ADD_RESOURCE(
  SYSTEM$REFERENCE('database', 'na_finance_db', 'SESSION', 'applybudget'));

CALL na_finance_budget!ADD_RESOURCE(
  SYSTEM$REFERENCE('warehouse', 'na_finance_wh', 'SESSION', 'applybudget'));
```

Grant instance roles to the custom roles you created in a previous step.

- Grant the required roles and privileges to the `budget_admin` role to let the `budget_admin` role modify and monitor the custom budget `na_finance_budget`:
- Grant the VIEWER instance role to the `budget_monitor` role to let the `budget_monitor` role monitor the custom budget `na_finance_budget`:

```
USE ROLE budget_owner;

GRANT SNOWFLAKE.CORE.BUDGET ROLE budgets_db.budgets_schema.na_finance_budget!ADMIN
  TO ROLE budget_admin;
GRANT SNOWFLAKE.CORE.BUDGET ROLE budgets_db.budgets_schema.na_finance_budget!VIEWER
  TO ROLE budget_monitor;
```

Monitoring credit usage

You have completed all the steps in the tutorial to activate your account budget, create a custom budget, and create custom roles to monitor and manage both account and custom budgets. Credit usage data for your budgets takes some time to populate.

Budgets uses serverless tasks to collect credit usage data for the budgets in your account. After you activate the account budget or create a custom budget, it takes a while for the serverless task to execute. After credit usage data becomes available, you can monitor credit usage for budgets using Snowsight.

To monitor credit usage after usage data becomes available, use the following steps:

Use the `account_budget_monitor` role created in a previous step and view the spending history for the account budget in the past week by executing the following statements:

```
USE ROLE account_budget_monitor;

CALL snowflake.local.account_root_budget!GET_SPENDING_HISTORY(
  TIME_LOWER_BOUND => DATEADD('days', -7, CURRENT_TIMESTAMP()),
  TIME_UPPER_BOUND => CURRENT_TIMESTAMP()
);
```

You can monitor spending history by service type. To view the spending history for the search optimization serverless feature for the account budget in the past week, execute the following statement:

```
USE ROLE account_budget_monitor;

CALL snowflake.local.account_root_budget!GET_SERVICE_TYPE_USAGE(
   SERVICE_TYPE => 'SEARCH_OPTIMIZATION',
   TIME_DEPART => 'day',
   USER_TIMEZONE => 'UTC',
   TIME_LOWER_BOUND => DATEADD('day', -7, CURRENT_TIMESTAMP()),
   TIME_UPPER_BOUND => CURRENT_TIMESTAMP()
);
```

Copy
Use the `budget_monitor` role to view the spending history for the past week for custom budget `na_finance_budget`:

```
USE ROLE budget_monitor;

CALL budgets_db.budgets_schema.na_finance_budget!GET_SPENDING_HISTORY(
  TIME_LOWER_BOUND => DATEADD('days', -7, CURRENT_TIMESTAMP()),
  TIME_UPPER_BOUND => CURRENT_TIMESTAMP()
);
```