



Teams GCPコスト情報連携の自動化

モーニングLT #4

2022年11月15日



株式会社ビッグツリーテクノロジー & コンサルティング

東京オフィス(本社)

〒108-0073

東京都港区三田3-13-16 三田43MTビル12F

札幌オフィス

〒060-0004 北海道札幌市中央区北4条西6-1

毎日札幌会館9F

ベトナムオフィス

7th Floor, Mercury Building, No.444 Hoang
Hoa Tham Street, Thuy Khue ward, Tay Ho
District, Hanoi city

西日本オフィス

〒530-0001 大阪府大阪市北区梅田2-2-2

ヒルトンプラザウエストオフィスタワー19F

シリコンバレーインキュベーションセンター

3350 Scott Blvd. #29 Santa Clara, CA
95054

1. 自己紹介
2. システム要件・アーキテクチャ
3. 実装
4. 参考
5. 質疑応答

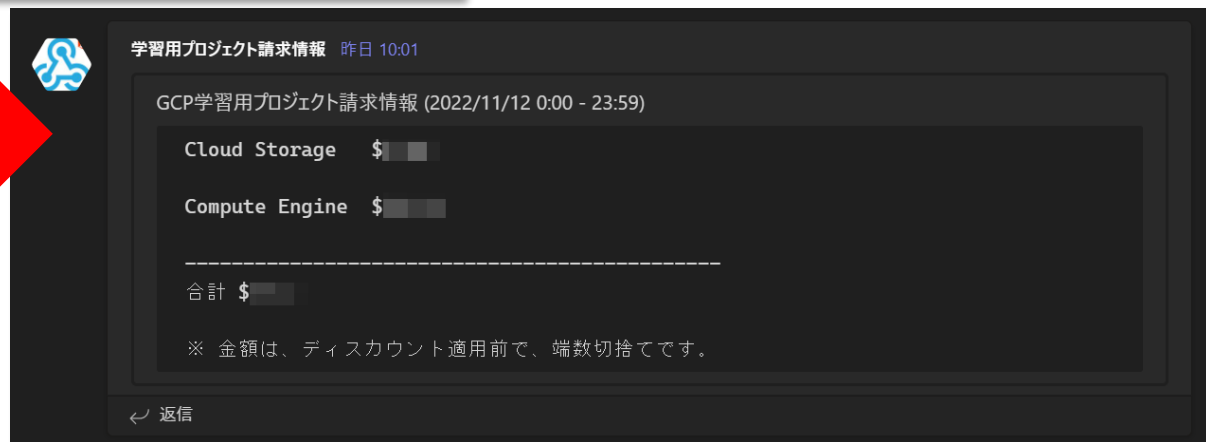
1.自己紹介

- ・ 長塚元規（ながつか もとき）
- ・ 株式会社ビックツリーテクノロジー & コンサルティング
 - <https://www.bigtreetc.com/>
- ・ DX事業部 / Cloud CoE
 - アパレルメーカーの業務システムの構築・運用保守
 - アプリケーション開発
 - クラウド案件の技術調査・検証の支援、実装
- ・ フロントエンド
 - React, Vue.js
- ・ バックエンド
 - Java, Python, Node.js, Go, PHP

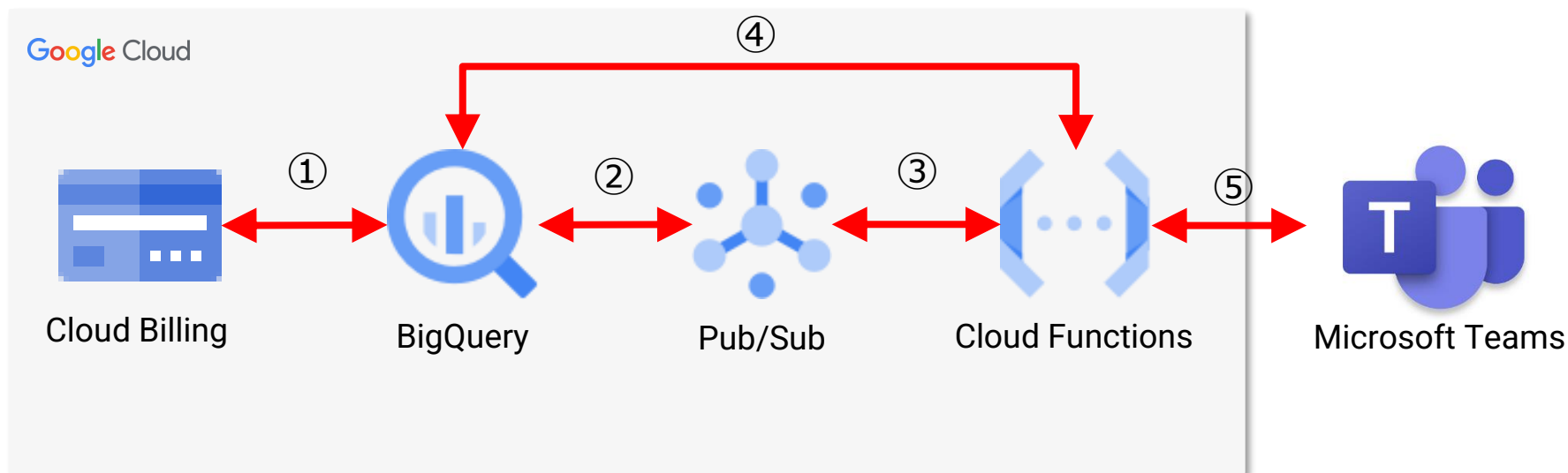


2.システム要件

- ・ 社内で運用しているGCPアカウントのコスト情報を管理するため、日次でTeamsの専用チャンネルに通知する
- ・ GCPリソース毎の利用料金の合計とリソースすべての利用料金の合計を表示



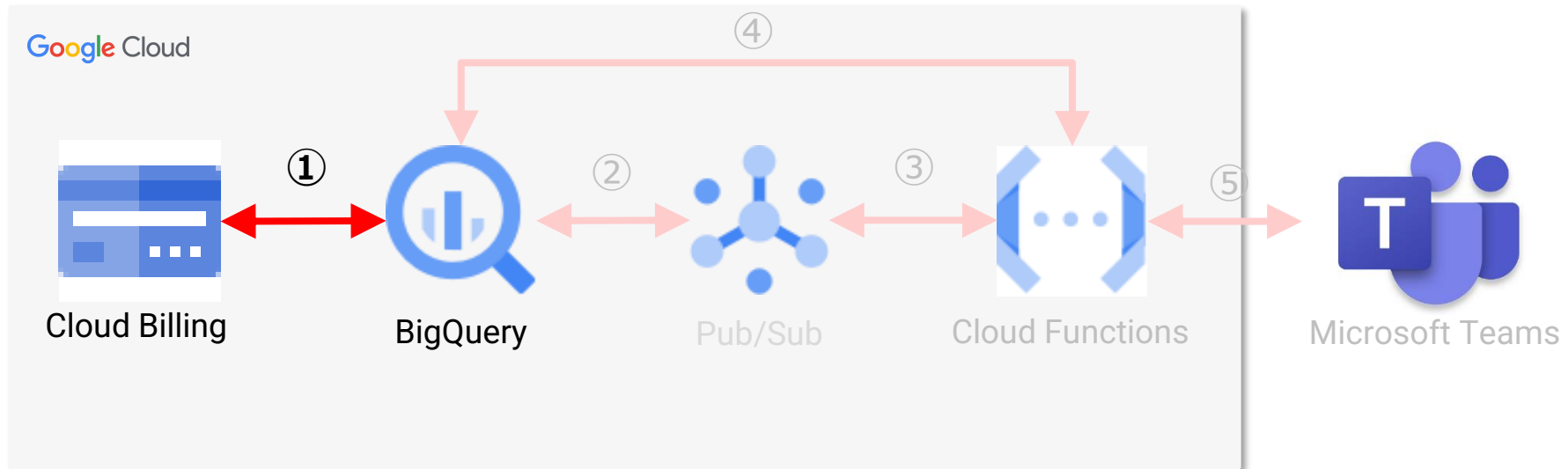
GCPのサーバレスサービスを利用してコスト情報のバッチ処理を実行する



- ① Cloud Billing データを BigQuery にエクスポートする
- ② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする
- ③ Pub/SubトピックをトリガーにCloud Functionsを起動する
- ④ Cloud FunctionsからBigQueryのコストデータを取得する
- ⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

3.実装① Billingデータを BigQuery にエクスポートする

GCPのサーバレスサービスを利用してコスト情報のバッチ処理を実行する



① Cloud Billing データを BigQuery にエクスポートする

② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする

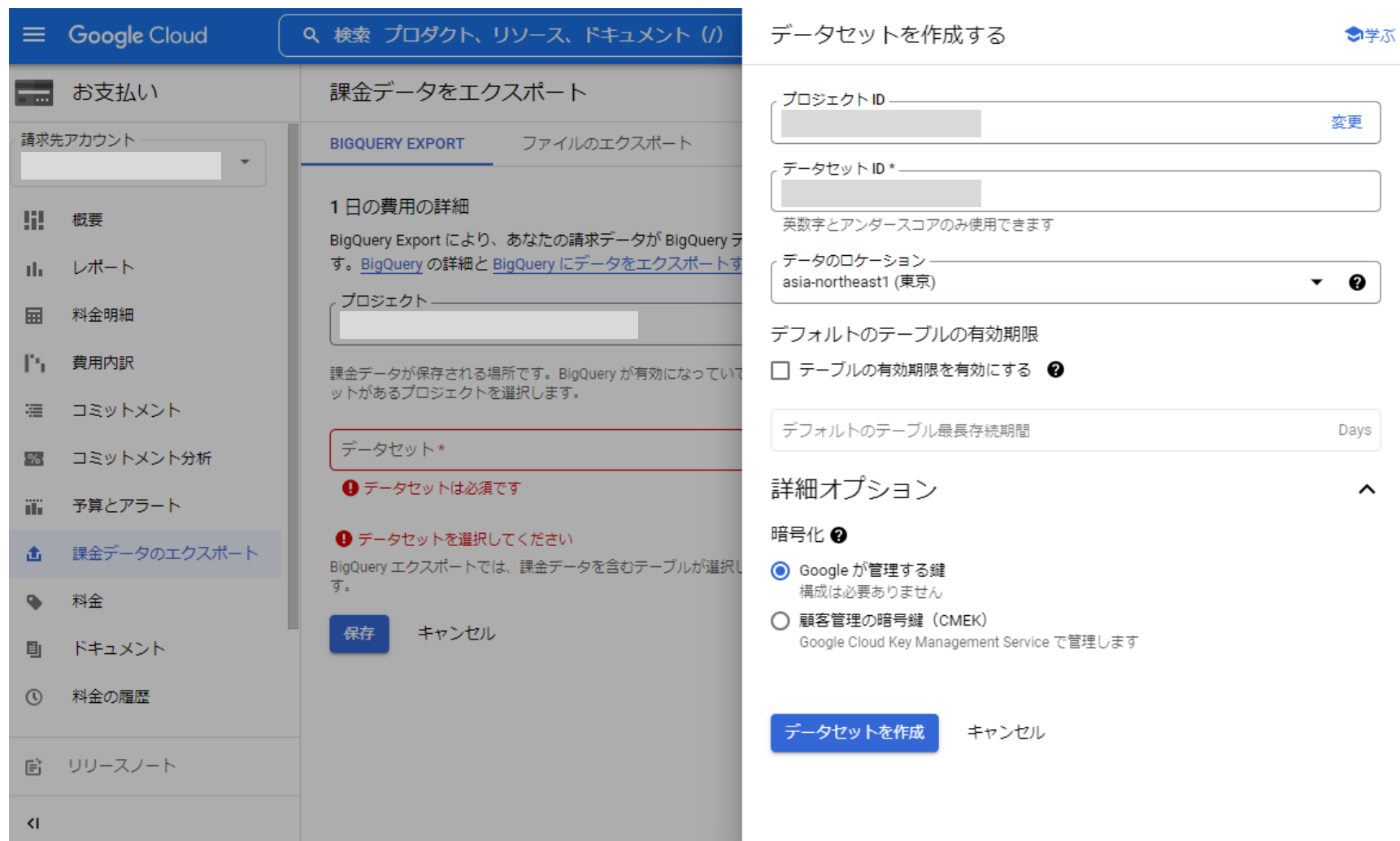
③ Pub/SubトピックをトリガーにCloud Functionsを起動する

④ Cloud FunctionsからBigQueryのコストデータを取得する

⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

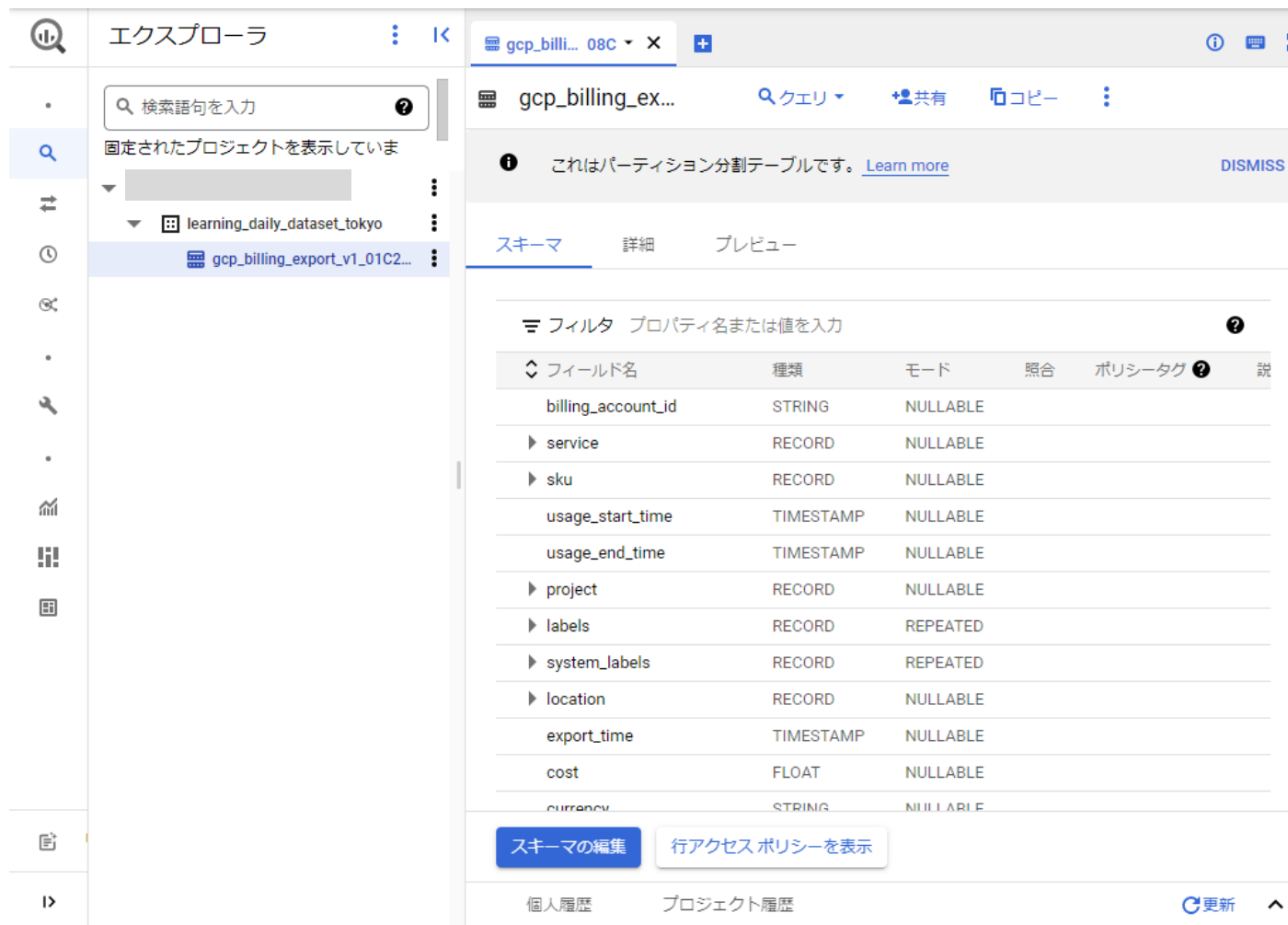
3.実装① Billingデータを BigQuery にエクスポートする

- 対象プロジェクトのBilling画面で「課金データのエクスポート」設定を行う



3.実装① Billingデータを BigQuery にエクスポートする

- BigQuery画面の対象プロジェクト配下にエクスポートしたデータセットが作成される
- スキーマは自動で構成される



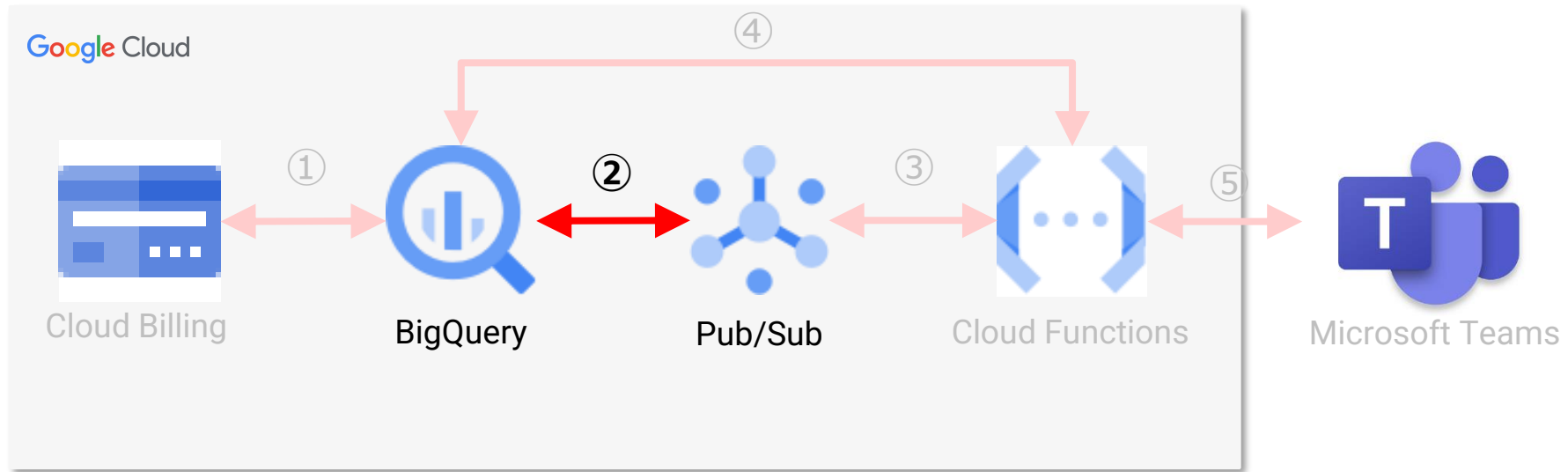
The screenshot shows the Google Cloud BigQuery Explorer interface. On the left, the 'Explorer' pane shows the project hierarchy: 'learning_daily_dataset_tokyo' > 'gcp_billing_export_v1_01C2...'. The main pane displays the schema for the 'gcp_billing_export_v1_01C2...' dataset. A message at the top indicates it is a partitioned table. The schema table lists fields with their names, types, and modes.

フィールド名	種類	モード	照合	ポリシータグ	説
billing_account_id	STRING	NULLABLE			
▶ service	RECORD	NULLABLE			
▶ sku	RECORD	NULLABLE			
usage_start_time	TIMESTAMP	NULLABLE			
usage_end_time	TIMESTAMP	NULLABLE			
▶ project	RECORD	NULLABLE			
▶ labels	RECORD	REPEATED			
▶ system_labels	RECORD	REPEATED			
▶ location	RECORD	NULLABLE			
export_time	TIMESTAMP	NULLABLE			
cost	FLOAT	NULLABLE			
currency	STRING	NULLABLE			

Buttons at the bottom: 'スキーマの編集' (Edit Schema), '行アクセスポリシーを表示' (Show Row Access Policy), '個人履歴' (Personal History), 'プロジェクト履歴' (Project History), and '更新' (Refresh).

3.実装② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする

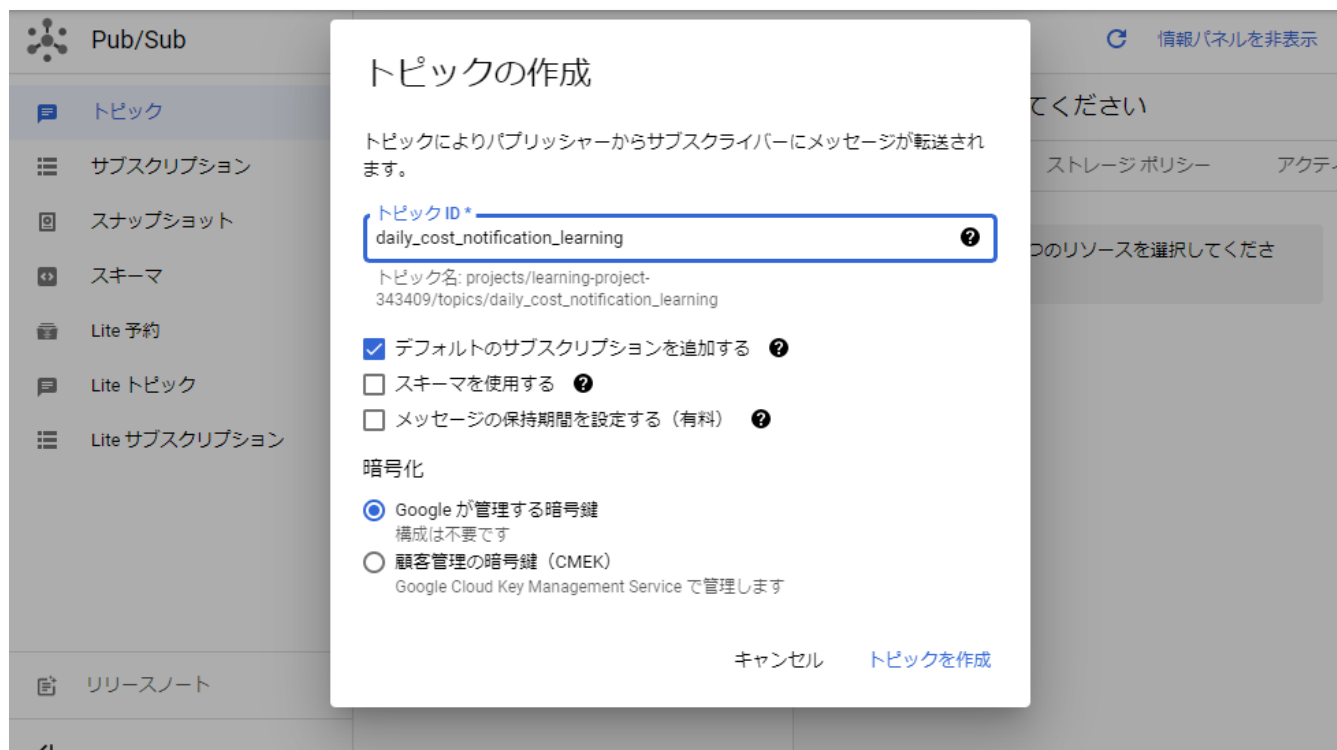
GCPのサーバレスサービスを利用してコスト情報のバッチ処理を実行する



- ① Cloud Billing データを BigQuery にエクスポートする
- ② **BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする**
- ③ Pub/SubトピックをトリガーにCloud Functionsを起動する
- ④ Cloud FunctionsからBigQueryのコストデータを取得する
- ⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

3.実装② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする

- Pub/Subトピックを新規作成する（Push）



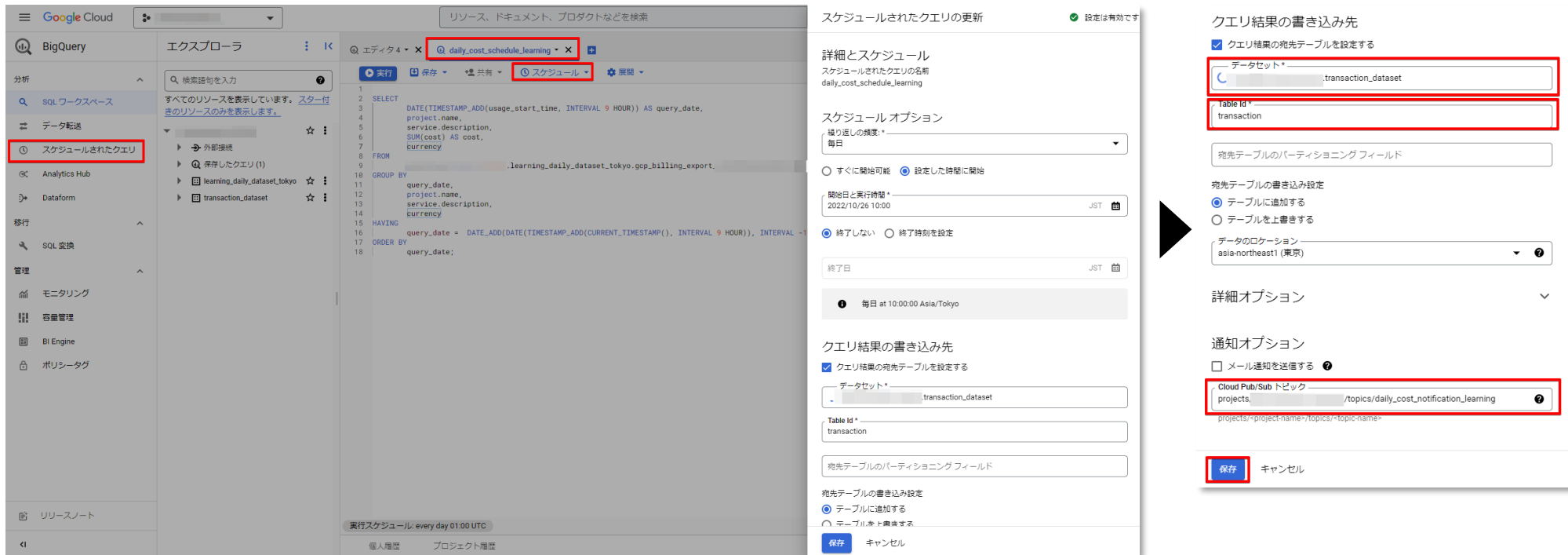
スケジュールされたクエリ [+ スケジュール設定されたクエリをエディタで作成](#) [削除](#)

フィルタ 転送構成のフィルタ処理

<input type="checkbox"/>	<input checked="" type="radio"/>	表示名	ソース	スケジュール (UTC)	リージョン	宛先データセット	今回のスケジュール UTC+9	↑
<input type="checkbox"/>	<input checked="" type="radio"/>	daily_cost_schedule_learning			asia-northeast1			

3.実装② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする

- 作成したデータセットに対してスケジュールクエリを作成する
- 作成したPub/Subトピックを通知先として設定する



The screenshot displays the Google Cloud BigQuery console interface. On the left, the 'BigQuery' menu is open, and 'スケジュールされたクエリ' (Scheduled queries) is highlighted. The main panel shows a SQL query editor with a query named 'daily_cost_schedule_learning'. The query is a SELECT statement that calculates the cost of transactions over time, grouped by date and project name. The query is scheduled to run every day at 10:00 UTC.

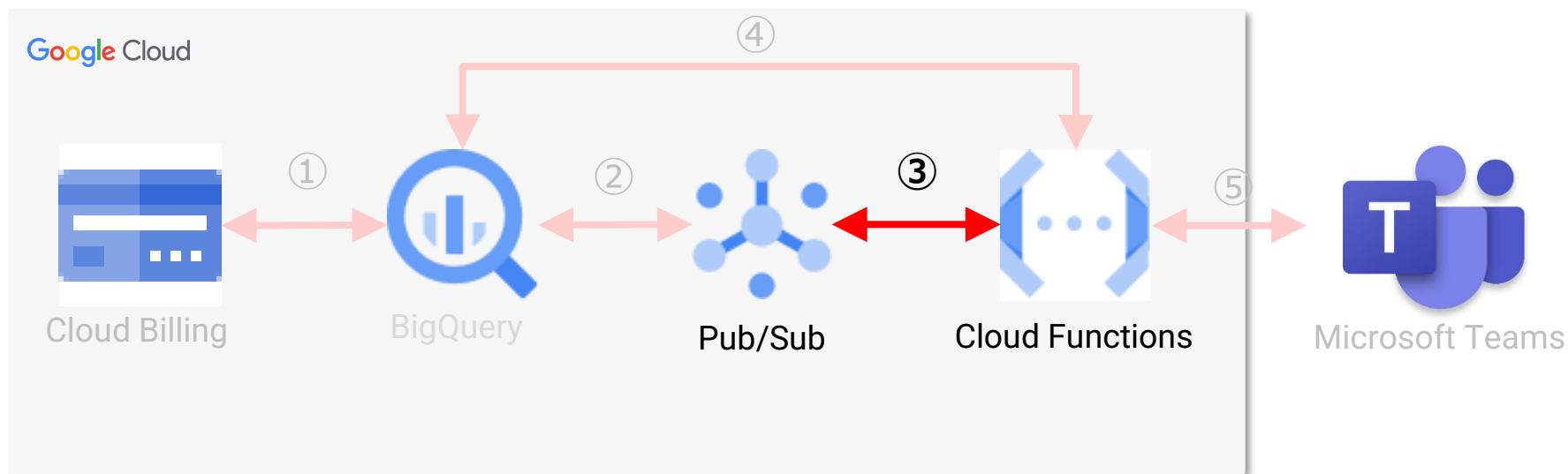
On the right, the 'スケジュールされたクエリの更新' (Update scheduled query) dialog is open. The 'クエリ結果の書き込み先' (Destination for query results) section is highlighted, showing the 'transaction_dataset' as the destination. The 'Table Id' is set to 'transaction'.

Below this, the '通知オプション' (Notification options) section is shown, where the 'Cloud Pub/Sub トピック' (Cloud Pub/Sub topic) is selected as the notification destination. The topic name is 'projects/<project-name>/topics/daily_cost_notification_learning'.

The '保存' (Save) button is highlighted in red, indicating the final step in the configuration process.

3.実装③ Pub/SubトピックをトリガーにCloud Functionsを起動する

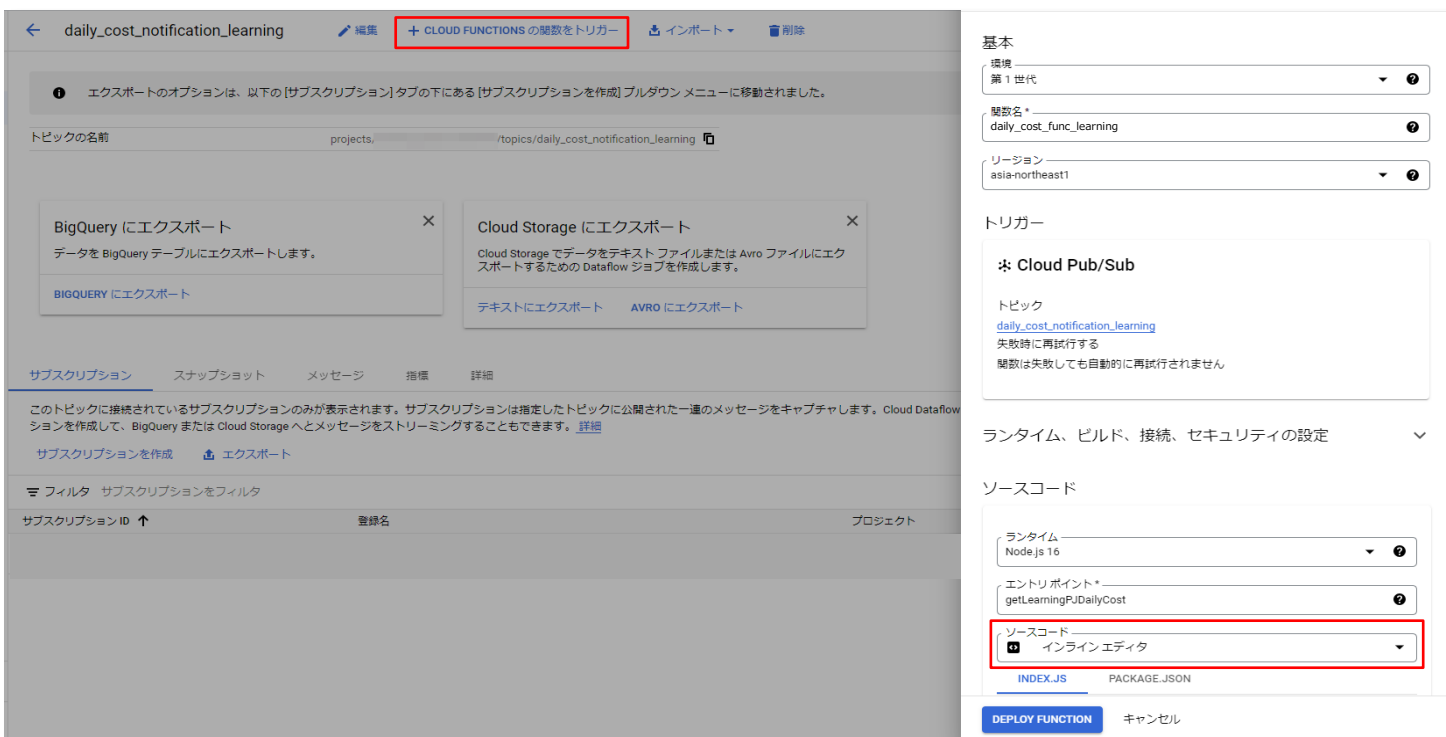
GCPのサーバレスサービスを利用してコスト情報のバッチ処理を実行する



- ① Cloud Billing データを BigQuery にエクスポートする
- ② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする
- ③ **Pub/SubトピックをトリガーにCloud Functionsを起動する**
- ④ Cloud FunctionsからBigQueryのコストデータを取得する
- ⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

3.実装③ Pub/SubトピックをトリガーにCloud Functionsを起動する

- Pub/Subトピック画面からトリガーするCloud Functionsを作成する
- index.jsとpackage.jsonは任意の方法でデプロイする（ブラウザエディタ、Zipファイル、gcloud SDKなど）



The screenshot shows the Google Cloud Console interface for creating a new Cloud Function. The 'Trigger' section is highlighted, showing the 'Cloud Pub/Sub' trigger type and the topic 'daily_cost_notification_learning'. The 'Source code' section is also highlighted, showing the 'Inline editor' option selected for the source code.

Cloud Functions								
Functions								
フィルタ 関数を絞り込む								
<input type="checkbox"/>	環境	名前	最終デプロイ日	リージョン	トリガー	ランタイム	割り当てられたメモリ	実行済みの関数
<input type="checkbox"/>	1st gen	daily_cost_func_learning	2022/10/25 16:39:52	asia-northeast1	トピック: daily_cost_notification_learning	Node.js 16	256 MB	getLearningPJDailyCost

3.実装③ Pub/SubトピックをトリガーにCloud Functionsを起動する

- Cloud Functionsのソースコード（全体）

```
const { BigQuery } = require('@google-cloud/bigquery');
const IncomingWebhook = require('ms-teams-webhook').IncomingWebhook;
require('date-utils');

/**
 * Generic background Cloud Function to be triggered by Pub/Sub.
 *
 * @param {object} event The event payload.
 * @param {object} context The event metadata.
 */
exports.getSamplePJDDailyCost = ( async (event, context) => {
  // Bigqueryクライアントを初期化
  const projectId = 'XXXX-XXXX'; // gcp project id
  const bigquery = new BigQuery({
    projectId: projectId,
  });
  const datasetName = 'transaction_dataset';
  const tableName = 'transaction'
  const projectName = 'XXXX'; // gcp project name

  const query = `SELECT
    name as project,
    description as service,
    round(sum(cost * 100)) / 100 as cost
  FROM ${projectId}.${datasetName}.${tableName}
  WHERE 1=1
  and name = '${projectId}'
  and query_date = DATE_SUB(CURRENT_DATE('Asia/Tokyo'), INTERVAL 1
  DAY)
  and cost >= 0.01
  GROUP BY project, service
  ORDER BY project, service`;

  console.log(`query: ${query}`);

  const queryOptions = {
    query: query,
    useLegacySql: false,
  };

```

```
// BigQuery transactionデータを取得しTeamsアラートメッセージを作成
try{
  // Wait for the query to finish
  const [rows] = await bigquery.query(queryOptions);

  let bqResult = "";
  let total = 0;
  for (let i = 0; i < rows.length; i++) {
    const row = rows[i];

    bqResult += row.service + '¥t$' + row.cost + '¥n¥n';
    console.log(bqResult);
    total += row.cost;
  }
  bqResult += '-----' + '¥n'
  bqResult += '合計 $' + (Math.floor(total * Math.pow(10, 2)) / Math.pow(10, 2)) +
  '¥n';

  const dt = new Date();
  dt.setDate(dt.getDate() - 1);
  const yesterday = dt.toFormat("YYYY/MM/DD");

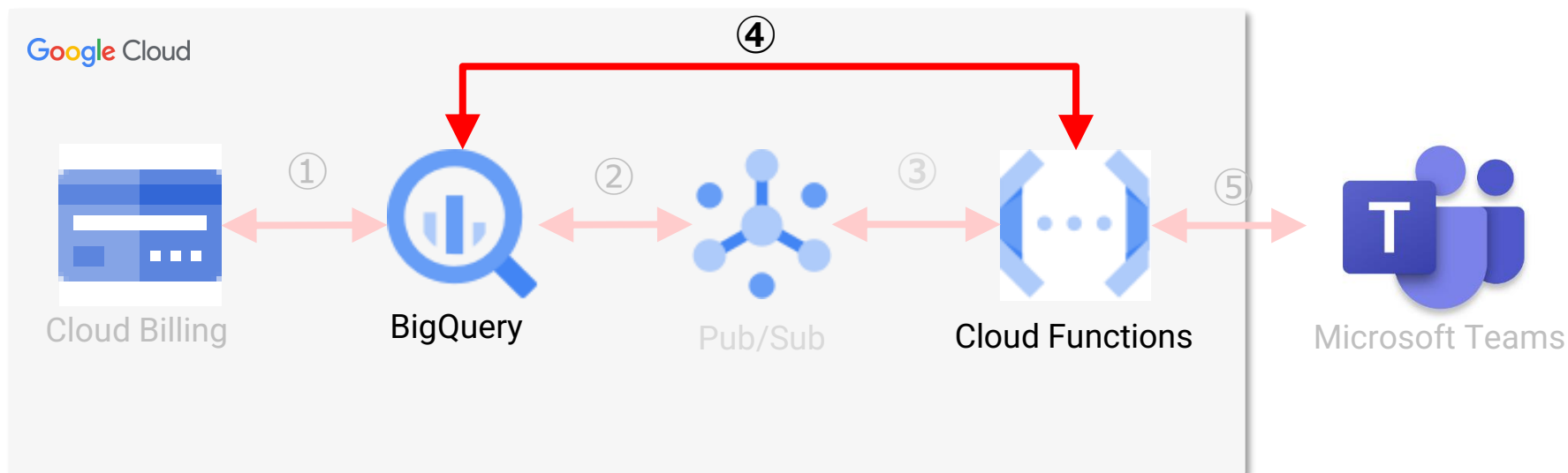
  // Teamsメッセージ作成
  let alertMessage = 'GCPXXXプロジェクト請求情報 (' + yesterday + ' 0:00 - 23:59)' +
  '¥n¥n';
  alertMessage += '````' + '¥n' + bqResult + '¥n';
  alertMessage += '※ 金額は、ディスカウント適用前で、端数切捨てです。';
  console.log(`alertMessage: ${alertMessage}`);

  // Teamsにメッセージ送信
  const url =
  'https://xxx.webhook.office.com/webhookb2/xxx/IncomingWebhook/xxx/xxx';
  const webhook = new IncomingWebhook(url);

  await webhook.send({
    text: alertMessage,
  });
  console.log('Temas Message transfer completed.')
} catch (error) {
  console.log(error);
}
});
```

3.実装④ Cloud FunctionsからBigQueryのコストデータを取得する

GCPのサーバレスサービスを利用してコスト情報のバッチ処理を実行する



- ① Cloud Billing データを BigQuery にエクスポートする
- ② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする
- ③ Pub/SubトピックをトリガーにCloud Functionsを起動する
- ④ Cloud FunctionsからBigQueryのコストデータを取得する**
- ⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

3.実装④ Cloud FunctionsからBigQueryのコストデータを取得する

- Cloud Functionsのソースコード（BigQueryデータ取得）

Node.js用の
BigQueryクライアントライブラリ

BigQueryクライアントを取得

BigQueryからコスト情報を
取得するクエリを作成

BigQueryからコスト情報を
取得するクエリを実行

```
const { BigQuery } = require('@google-cloud/bigquery');
const IncomingWebhook = require('ms-teams-webhook').IncomingWebhook;
require('date-utils');

/**
 * Generic background Cloud Function to be triggered by Pub/Sub.
 *
 * @param {object} event The event payload.
 * @param {object} context The event metadata.
 */
exports.getSamplePJDailyCost = ( async (event, context) => {
  // BigQueryクライアントを初期化
  const projectId = 'XXXX-XXXX'; // gcp project id
  const bigquery = new BigQuery({
    projectId: projectId,
  });
  const datasetName = 'transaction_dataset';
  const tableName = 'transaction'
  const projectName = 'XXXX'; // gcp project name

  const query = `SELECT
    name as project,
    description as service,
    round(sum(cost * 100)) / 100 as cost
  FROM ${projectId}.${datasetName}.${tableName}
  WHERE 1=1
    and name = '${projectName}'
    and query_date = DATE_SUB(CURRENT_DATE('Asia/Tokyo'), INTERVAL 1 DAY)
    and cost >= 0.01
  GROUP BY project, service
  ORDER BY project, service`;

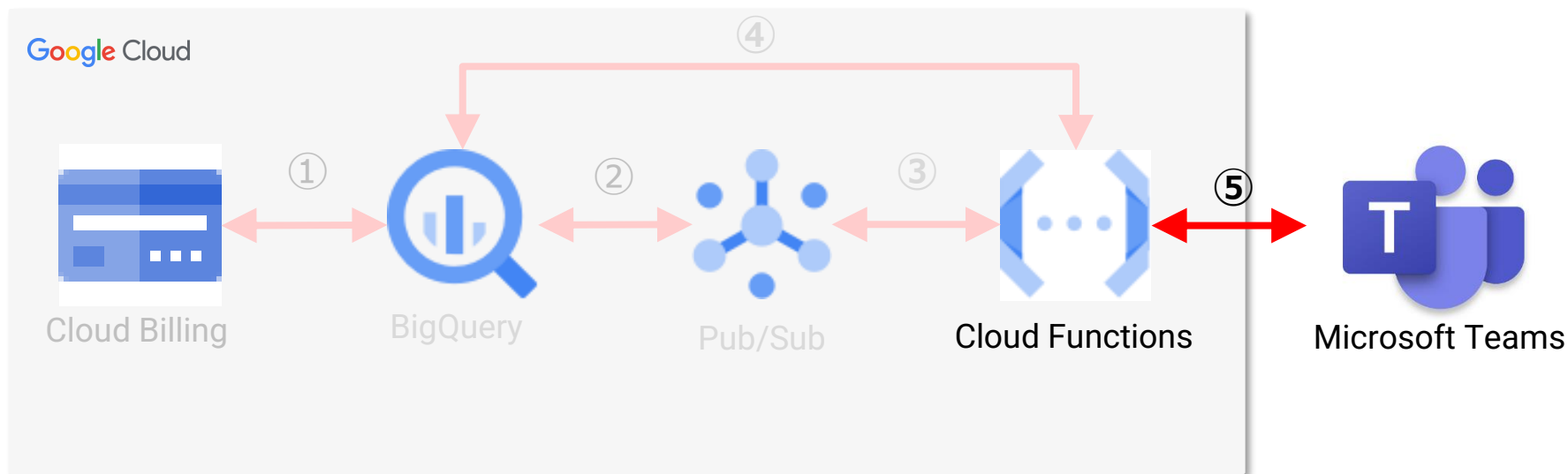
  console.log(`query: ${query}`);

  const queryOptions = {
    query: query,
    useLegacySql: false,
  };

  // BigQuery transactionデータを取得しTeamsアラートメッセージを作成
  try{
    // Wait for the query to finish
    const [rows] = await bigquery.query(queryOptions);
```


3.実装⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

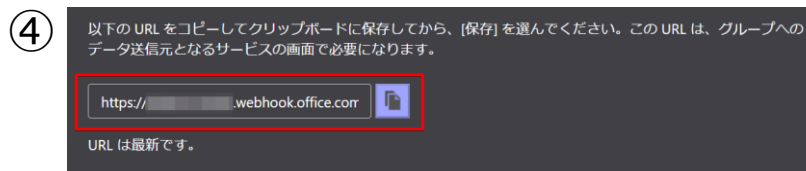
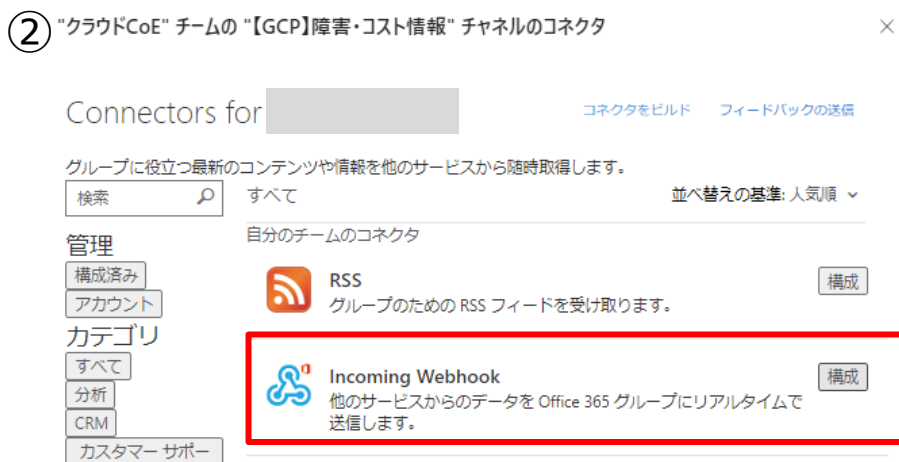
GCPのサーバレスサービスを利用してコスト情報のバッチ処理を実行する



- ① Cloud Billing データを BigQuery にエクスポートする
- ② BigQueryのスケジュールクエリをPub/Subトピックにパブリッシュする
- ③ Pub/SubトピックをトリガーにCloud Functionsを起動する
- ④ Cloud FunctionsからBigQueryのコストデータを取得する
- ⑤ **Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する**

3.実装⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

- Teamsで通知を行いたいチャンネルでIncoming Webhookを作成しURLを取得する



3.実装⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

- Cloud Functionsのソースコード（メッセージ投稿）

Node.js用の
Incoming Webhook
クライアントライブラリ

BigQueryデータの成形

メッセージの作成

メッセージの投稿

```
const IncomingWebhook = require('ms-teams-webhook').IncomingWebhook;

(中略)

try{
  const [rows] = await bigquery.query(queryOptions);

  let bqResult = "";
  let total = 0;
  for (let i = 0; i < rows.length; i++) {
    const row = rows[i];

    bqResult += row.service + '¥t$' + row.cost + '¥n¥n';
    console.log(bqResult);
    total += row.cost;
  }
  bqResult += '-----' + '¥n'
  bqResult += '合計 $' + (Math.floor(total * Math.pow(10, 2)) / Math.pow(10, 2)) + '¥n';

  const dt = new Date();
  dt.setDate(dt.getDate() - 1);
  const yesterday = dt.toFormat("YYYY/MM/DD");

  // Teamsメッセージ作成
  let alertMessage = 'GCPXXXプロジェクト請求情報 (' + yesterday + ' 0:00 - 23:59)' + '¥n¥n';
  alertMessage += '¥n' + '¥n' + bqResult + '¥n';
  alertMessage += '※ 金額は、ディスカウント適用前で、端数切捨てです。';
  console.log(`alertMessage: ${alertMessage}`);

  // Teamsにメッセージ送信
  const url = 'https://xxx.webhook.office.com/webhookb2/xxx/IncomingWebhook/xxx/xxx';
  const webhook = new IncomingWebhook(url);

  await webhook.send({
    text: alertMessage,
  });
  console.log('Temas Message transfer completed.')
} catch (error) {
  console.log(error);
}
});
```

3.実装⑤ Cloud FunctionsからTeamsのチャンネルにメッセージを投稿する

- BigQueryのクエリが定期実行され、Pub/Subトピックを介してCloud Functionsが起動し、メッセージがTeamsのチャンネルに投稿される

①

スケジュールされたクエリの詳細

daily_cost_schedule_learning

スケジュール (UTC) 次の実行の予定日
every day 01:00 2022年11月15日 10:00:00 UTC+9

フィルタ 転送実行のフィルタ

	実行日	実行予定時間 UTC+9	概要
○ ✓	2022年11月14日	2022年11月14日 10:00:00 UTC+9	転送実行が正常に完了しました。
○ ✓	2022年11月13日	2022年11月13日 10:00:00 UTC+9	転送実行が正常に完了しました。
○ ✓	2022年11月12日	2022年11月12日 10:00:00 UTC+9	転送実行が正常に完了しました。

②

Cloud Functions 関数の詳細

daily_cost_func_learning 第1世代

バージョン: 48, デプロイ時刻: 2022/10/25 16:...

指標 詳細 ソース 変数 トリガー 権限 ログ テスト中

ログ 152 件のログエントリ

重大度	タイムスタンプ	概要
> 0	2022-11-13 10:01:17.575 JST	daily_cost_func_learning Function execution started
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning query: SELECT
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning name as project,
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning description as service,
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning round(sum(cost * 100)) / 100 as cost
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning FROM transaction_dataset.transaction
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning WHERE 1=1
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning and name = 'Learning Project'
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning and query_date = DATE_SUB(CURRENT_DATE('Asia/Tokyo'), INTERVAL 1 DAY)
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning and cost >= 0.01
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning GROUP BY project, service
> *	2022-11-13 10:01:18.922 JST	daily_cost_func_learning ORDER BY project, service
> *	2022-11-13 10:01:19.856 JST	daily_cost_func_learning Cloud Storage \$
> *	2022-11-13 10:01:19.856 JST	daily_cost_func_learning Cloud Storage \$
> *	2022-11-13 10:01:19.856 JST	daily_cost_func_learning Compute Engine \$
> *	2022-11-13 10:01:19.856 JST	daily_cost_func_learning alertMessage: GCP学習用プロジェクト請求情報 (2022/11/12 0:00 - 23:59)

③

学習用プロジェクト請求情報 昨日 10:01

GCP学習用プロジェクト請求情報 (2022/11/12 0:00 - 23:59)

Cloud Storage	\$
Compute Engine	\$
合計	\$

※ 金額は、ディスカウント適用前で、端数切捨てです。

返信

- 今回の発表で作成したCloud Functionsのサンプルソースコード
 - https://github.com/btc-nagatsuka/gcp_daily_cost_notification_teams
- (公式) BigQuery への課金データのエクスポートで GCP 利用コストを管理
 - <https://cloud.google.com/blog/ja/products/gcp/monitor-and-manage-your-costs-with>
- (公式) Node.jsでの依存関係の指定
 - package.jsonをCloud Functionsにアップロードすることで外部モジュールの使用が可能です
 - <https://cloud.google.com/functions/docs/writing/specifying-dependencies-nodejs?hl=ja>
- (公式) ローカルマシンからデプロイする
 - ローカル環境からGCP上にCloud Functionsの資源をデプロイする方法です
 - <https://cloud.google.com/functions/docs/deploy?hl=ja#from-local-machine>
- Cloud Pub/Sub + Node.jsのローカル開発環境の構築
 - 今回は利用していませんがローカルでPub/Subを実行できるエミュレータがあるようです
 - <https://zenn.dev/sykmhmf/articles/33d7b293dc5e45>
- (公式) Functions Framework
 - 今回は利用していませんがローカルでCloud Functionsを実行できるエミュレータがあるようです
 - <https://cloud.google.com/functions/docs/functions-framework?hl=ja>

ご清聴ありがとうございました！