



IoT ハンズオン Azure IoT 基礎～Power BI 可視化 実践編

日本マイクロソフト株式会社

目次

- **概要**

1. サンプルプログラムにて生成したデータをAzure IoT Hub経由でAzureへ渡す
2. 届いたJSONデータをAzure Stream Analyticsで解析してストレージに保存
3. ストレージに溜まったデータをAzure Synapse Analyticsを利用して参照
4. Power BIを用いたデータの可視化
5. 収集データがある閾値を越えた場合にTeamsへアラートを出す

- **事前準備**

- Python 環境の準備
- Power BI Desktop のインストール
- サンプルコードのダウンロード

IoT 実現のためのステップ

モノがインターネットにつながる

センサーの取り付けやスマートデバイスの開発など、既存の設備や資産からデータを取得するためのシステムを構築します。



もの
Things

モノからデータを取得

設備や資産を制御、監視、管理する「IoT ソリューション」を導入することで、リアルタイムデータを取得できるようになります。



洞察
Insights

データを活用し自動化／分析

収集したデータを高度に分析することで、いままでは気づかなかつた新たなビジネスインサイト（洞察）を獲得します。



アクションを起こす

強力なアプリケーションを使ってインサイトを実践に移し、新たな収益とビジネスチャンスを創出します。



行動/カイゼン
Actions

IoT + Analytics + Action

データ収集・蓄積



Things

工場データをクラウドに
収集し、蓄積する

分析



Insights

AI等を適用し
要因探索や予測をする

業務適用

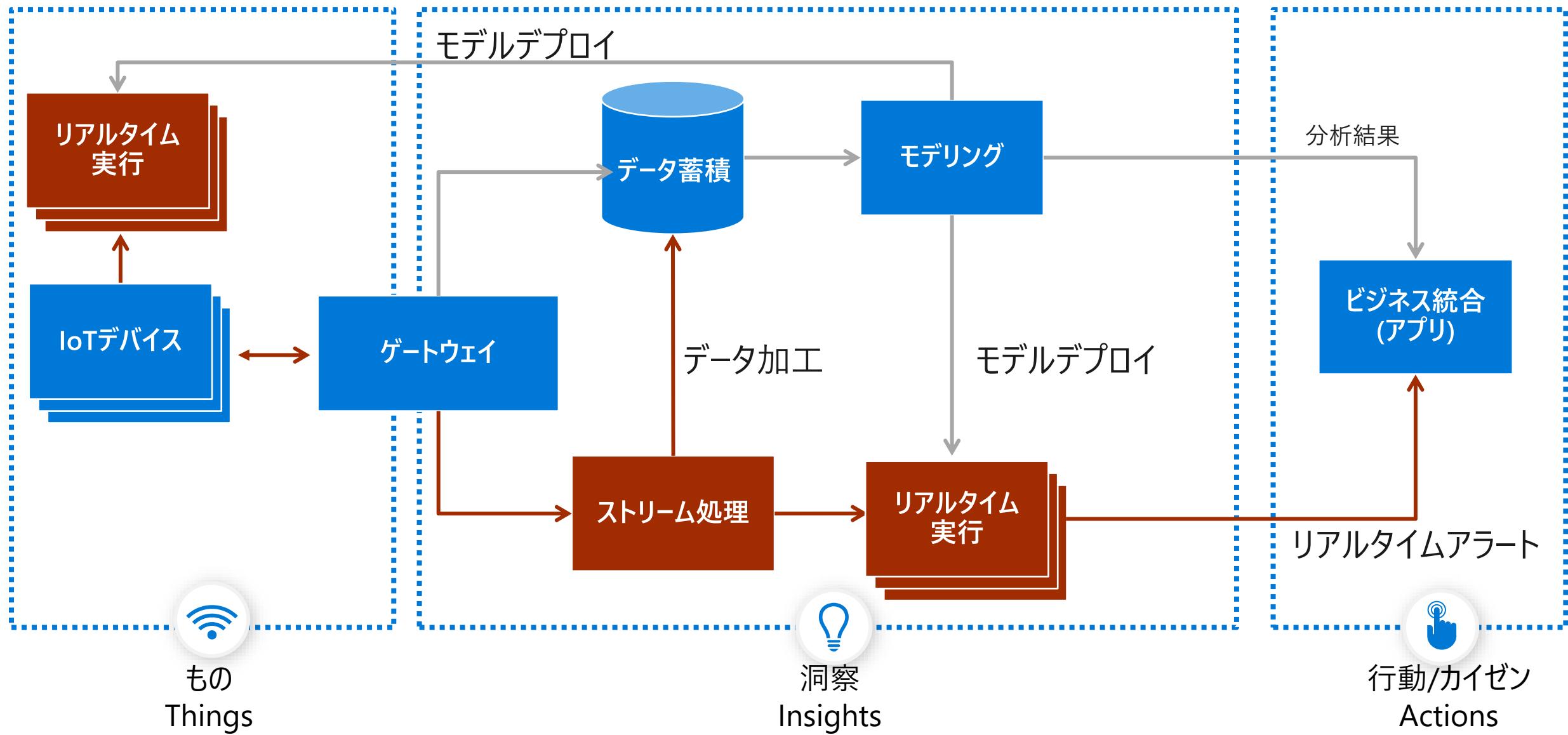


Actions

解析結果の可視化、
業務プロセスへの適用

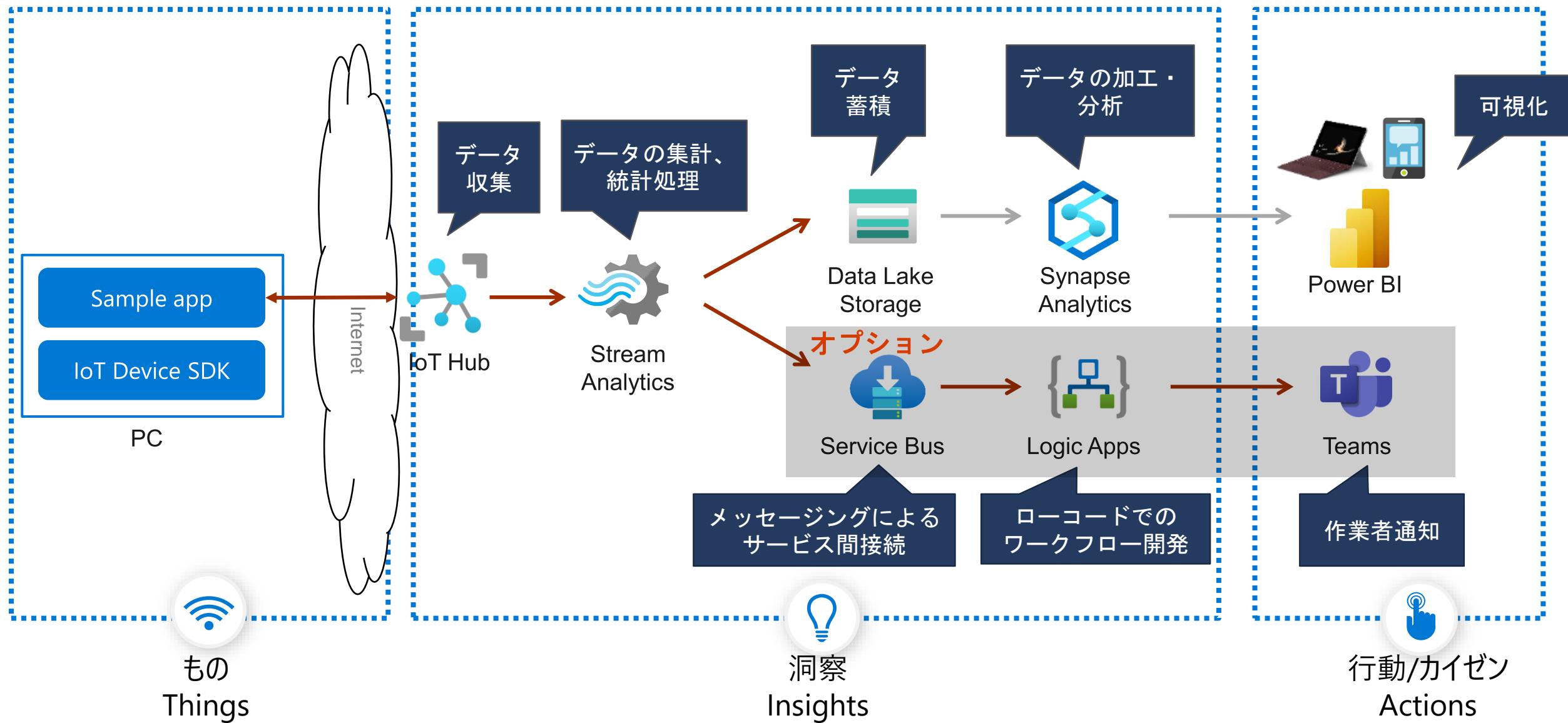
IoT リファレンスアーキテクチャ

→ バッチ
→ リアルタイム



ハンズオン構成

→ バッチ
→ リアルタイム

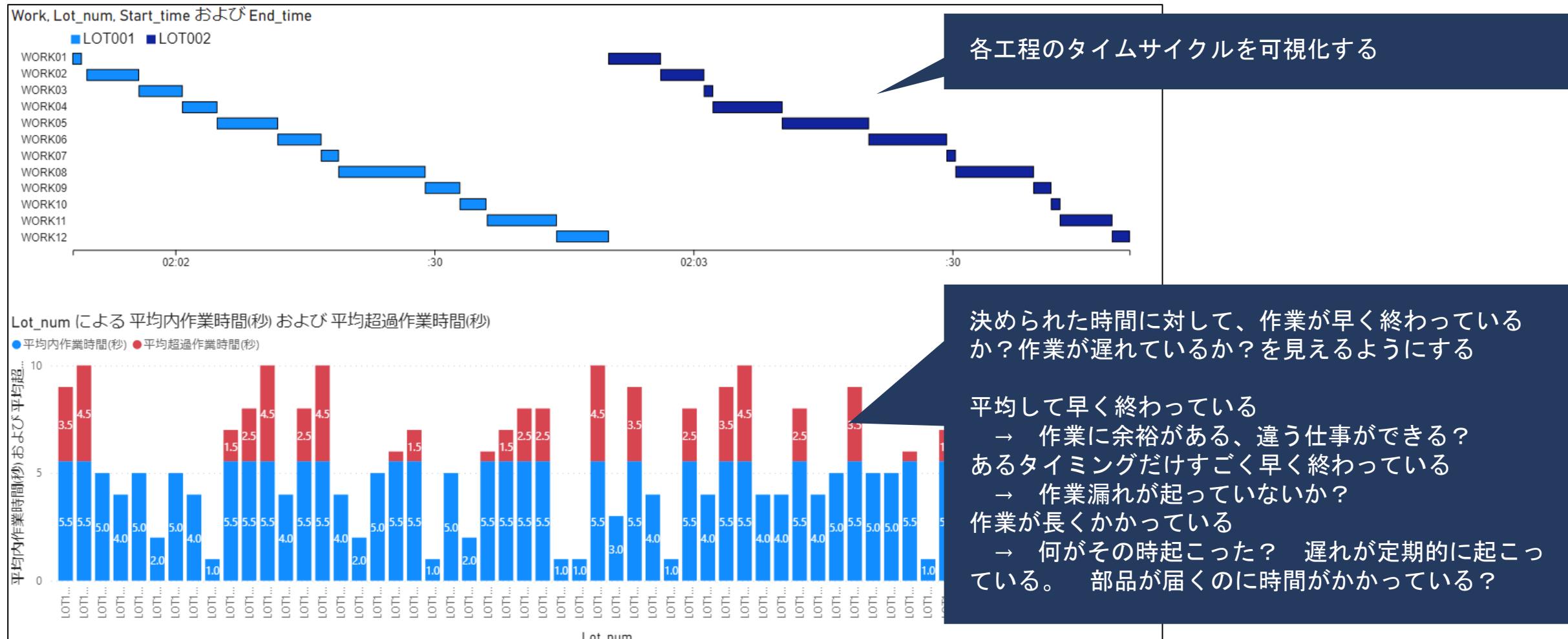


ハンズオンの流れ

1. IoT Hub の作成
2. サンプルプログラムの実行
3. Synapse Analytics の作成
4. Stream Analytics の作成
5. Synapse Analytics による Query および外部テーブルの作成
6. Power BI レポートの作成
7. Teams アラート連携

サンプルレポート

本日のハンズオンで作成するサンプルレポートです



0. 事前準備

- Python環境の準備
- Power BI Desktop のインストール
- サンプルコードのダウンロード

1. Python 環境の準備

※既に Python 3 環境お持ちの場合、「Python のインストール」は不要です
「Python パッケージのインストール」のみ、事前に実施をお願いします

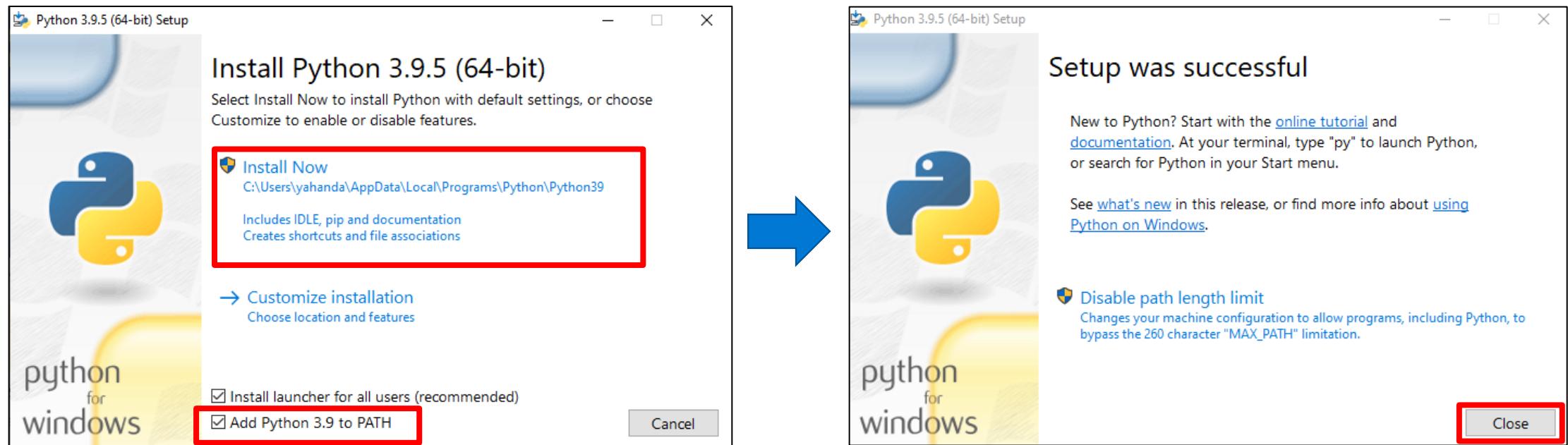
Python のインストール (1/2)

- 以下にアクセスし、Python 3.9.5 for Windows 64bit版のインストーラーをダウンロード
- <https://www.python.org/downloads/>



Python のインストール (2/2)

- インストーラーを起動し、「Add Python 3.9 to PATH」にチェックを入れて、「Install Now」をクリック
- 「Setup was successful」が表示されたら「Close」をクリック



Python パッケージのインストール

- ・ コマンドプロンプトを立ち上げて、以下を実行します
 - ・ pip install azure-iot-device
- ・ 参考：Azure IoT Devide SDK (python)
 - ・ <https://github.com/Azure/azure-iot-sdk-python/tree/master/azure-iot-device#installation>

```
C:\> Command Prompt
Microsoft Windows [Version 10.0.19043.1052]
(c) Microsoft Corporation. All rights reserved.

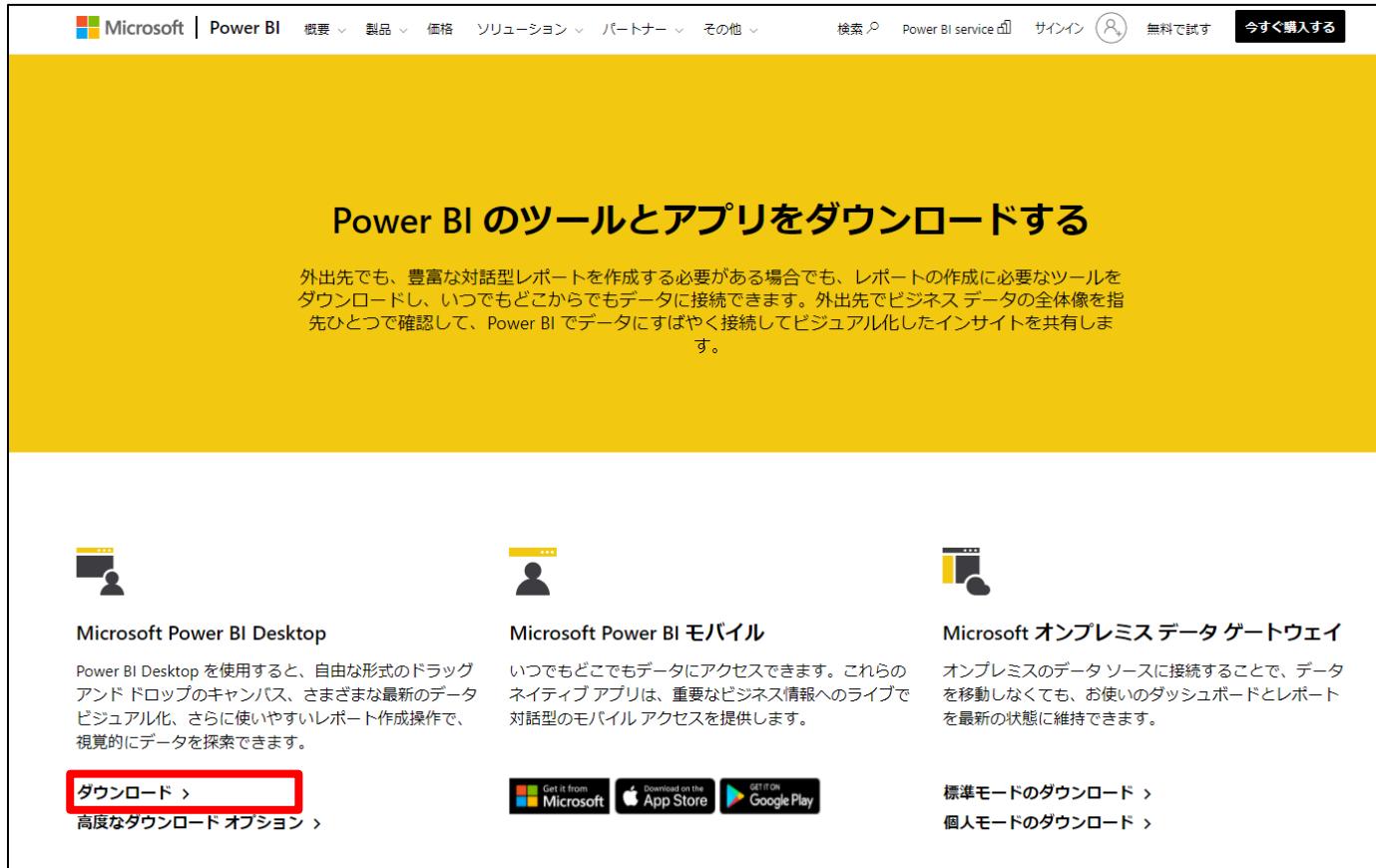
C:\$Users\$yahanda>pip install azure-iot-device
Collecting azure-iot-device
  Downloading azure_iot_device-2.7.0-py2.py3-none-any.whl (162 kB)
|████████████████████████████████████████████████████████████████████████████████| 162 kB 1.1 MB/s
Collecting six<2.0.0,>=1.12.0
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting paho-mqtt<2.0.0,>=1.4.0
  Using cached paho-mqtt-1.5.1.tar.gz (101 kB)
Collecting requests-unixsocket<1.0.0,>=0.1.5
  Using cached requests_unixsocket-0.2.0-py2.py3-none-any.whl (11 kB)
Collecting janus==0.4.0
  Using cached janus-0.4.0-py3-none-any.whl (7.0 kB)
Collecting PySocks
  Using cached PySocks-1.7.1-py3-none-any.whl (16 kB)
Collecting deprecation<3.0.0,>=2.1.0
  Using cached deprecation-2.1.0-py2.py3-none-any.whl (11 kB)
Collecting requests<3.0.0,>=2.20.0
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting urllib3<1.27,>=1.26.5
  Downloading urllib3-1.26.5-py2.py3-none-any.whl (138 kB)
|████████████████████████████████████████████████████████████████████████████████| 138 kB 1.1 MB/s
Collecting packaging
  Downloading packaging-20.9-py2.py3-none-any.whl (40 kB)
|████████████████████████████████████████████████████████████████████████████████| 40 kB 2.6 MB/s
Collecting certifi>=2017.4.17
  Downloading certifi-2021.5.30-py2.py3-none-any.whl (145 kB)
|████████████████████████████████████████████████████████████████████████████████| 145 kB 6.4 MB/s
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting chardet<5,>=3.0.2
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting pyparsing>=2.0.2
  Using cached pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
Using legacy 'setup.py install' for paho-mqtt, since package 'wheel' is not installed.
Installing collected packages: urllib3, pyparsing, idna, chardet, certifi, requests, packaging, paho-mqtt, janus, deprecation, azure-iot-device
  Running setup.py install for paho-mqtt ... done
Successfully installed PySocks-1.7.1 azure-iot-device-2.7.0 certifi-2021.5.30 chardet-4.0.0 janus-0.4.0 packaging-20.9 paho-mqtt-1.5.1 pyparsing-2.4.7 requests-2.25.1 requests-unixsocket-1.26.5
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the 'c:\$users\$yahanda\$appdata\$local\$programs\$python\$python39\Scripts\python.exe --upgrade pip' command.

C:\$Users\$yahanda>
```

2. Power BI Desktop のインストール

Power BI Desktop のインストール

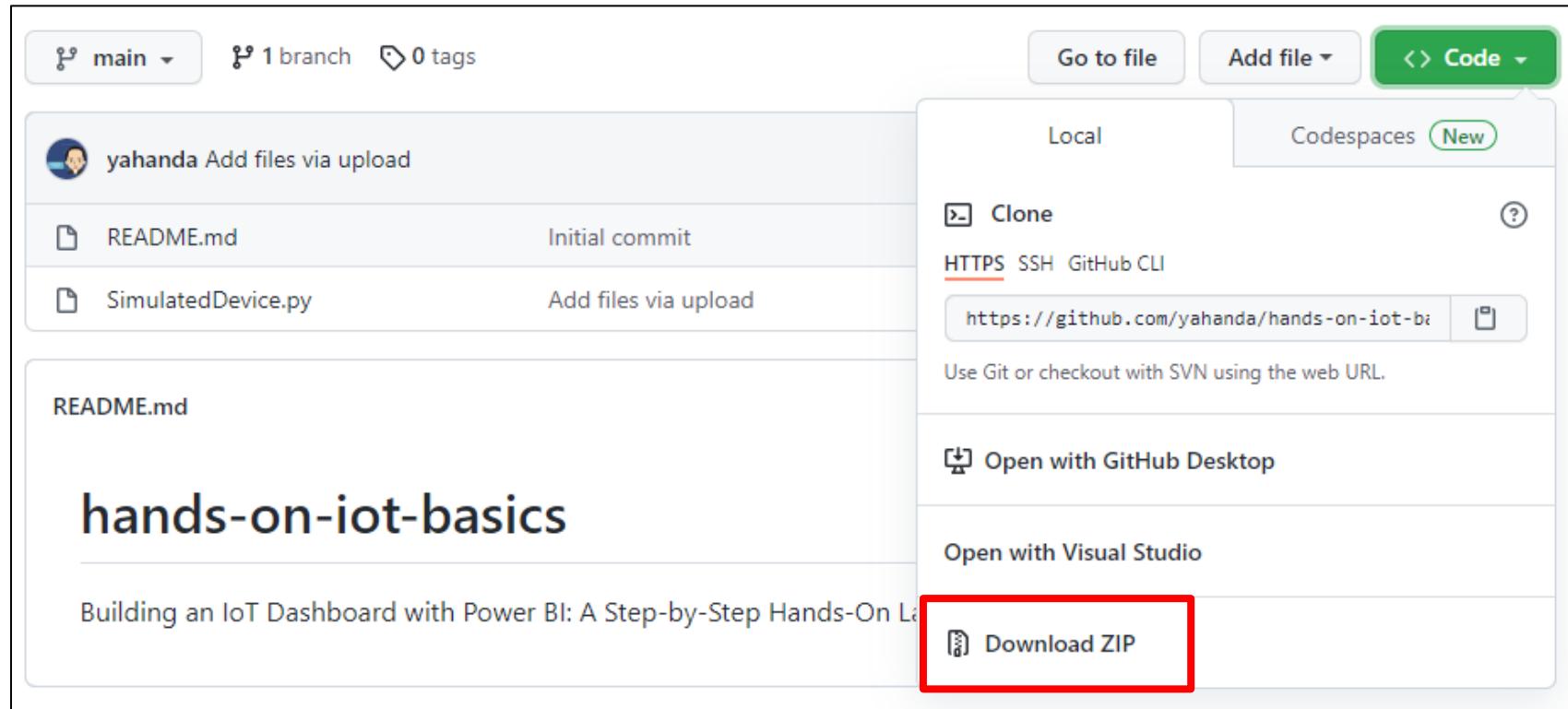
- 以下にアクセスし「ダウンロード」をクリックし、Microsoft Storeからインストールください
- <https://powerbi.microsoft.com/ja-jp/downloads/>



3. サンプルコードのダウンロード

サンプルコードのダウンロード

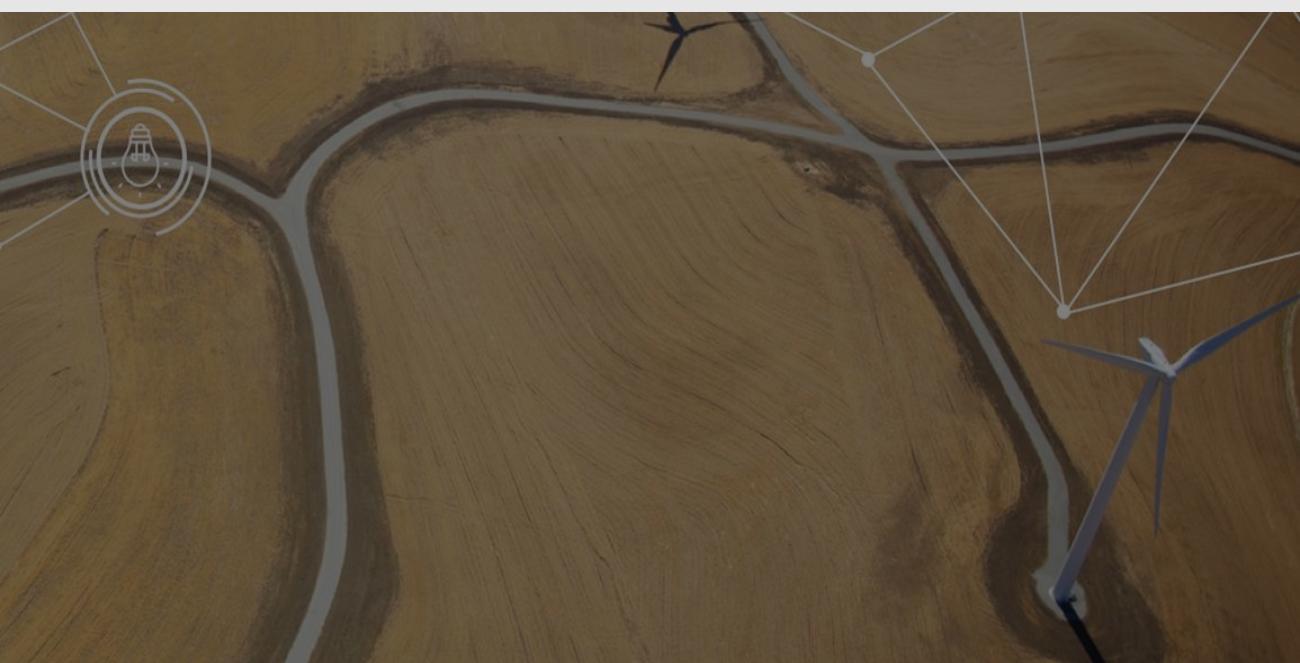
- ・ ハンズオンで使用するpythonのサンプルコード (SimulatedDevice.py) を、GitHubで公開しておりますので、
使用するPCの任意のフォルダに格納ください
- ・ <https://github.com/yahanda/hands-on-iot-basics>



1. IoT Hub の作成



Azure IoT Hub



数十億台の IoT デバイスとの双方向通信を確立



デバイスごとの認証によってセキュリティを強化



IoT Hub Device Provisioning Serviceを使用して大規模にデバイスをプロビジョニング



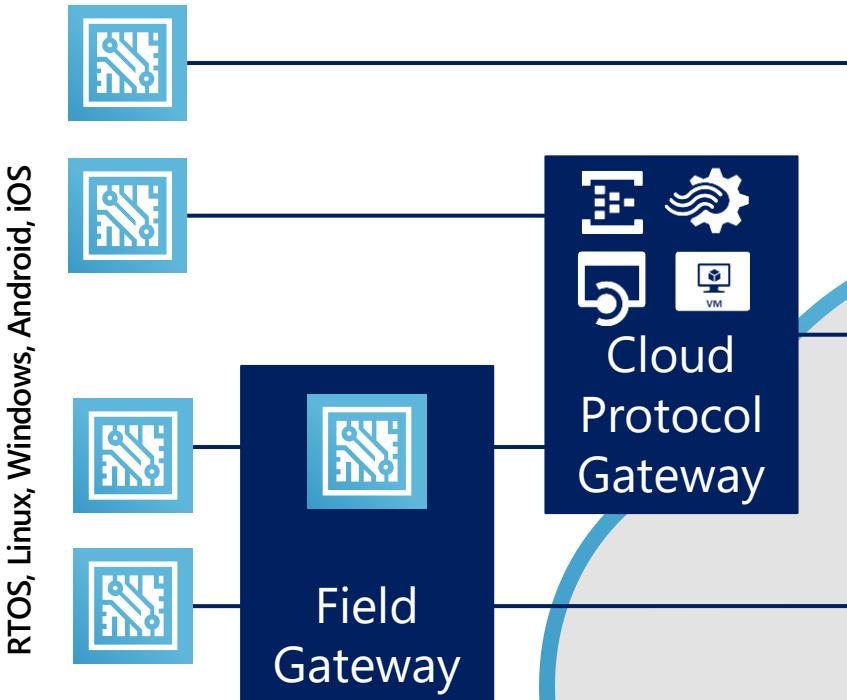
デバイス管理によって大規模にデバイスを管理



多言語でオープンソースの SDK

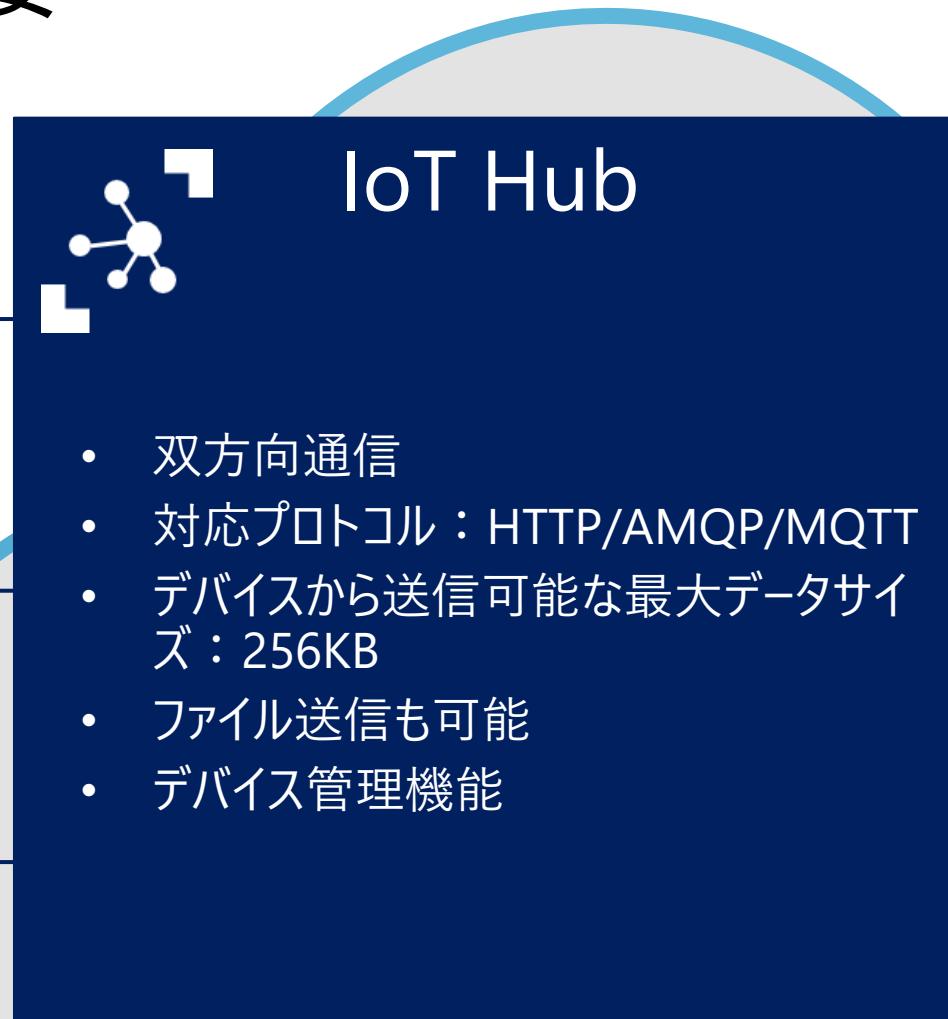
Azure IoT Hub 概要

Devices



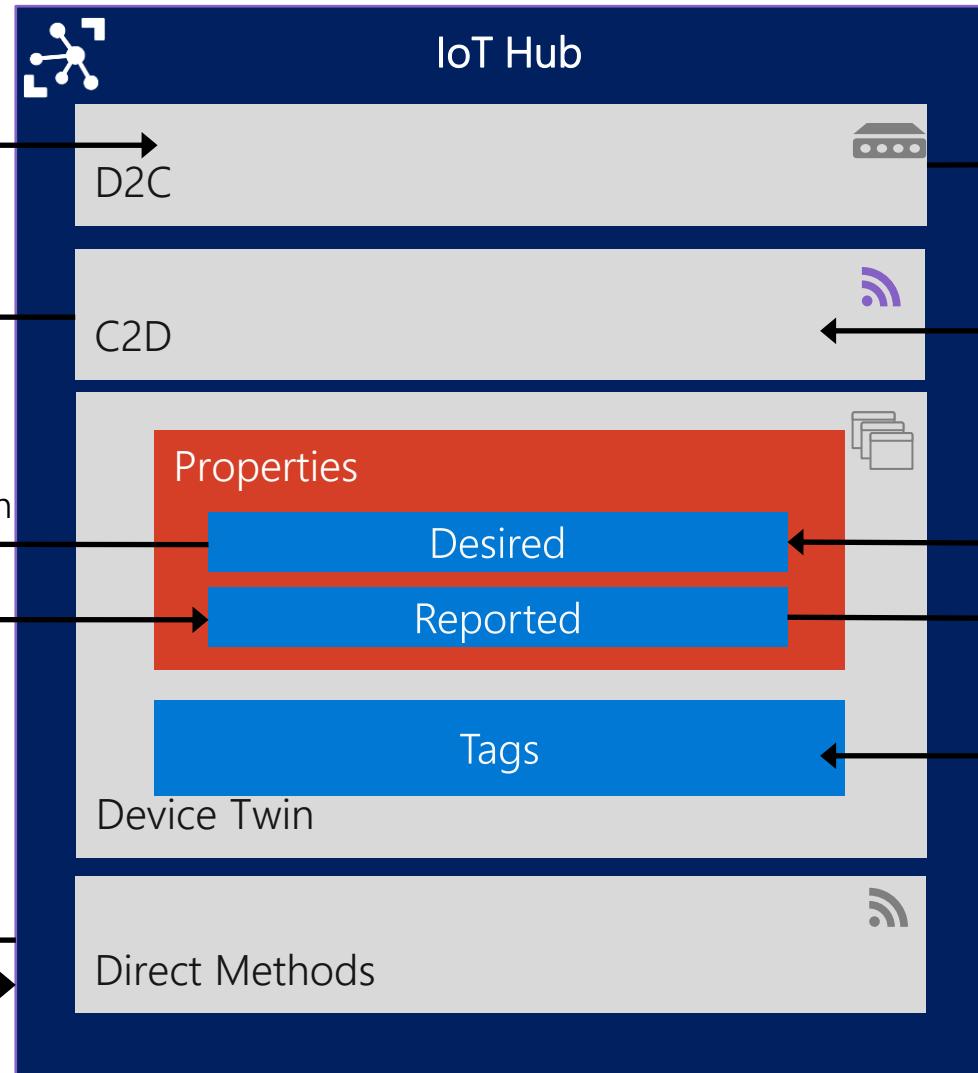
組込み機器向けSDK

- C#
- C/C++
- Java
- JavaScript
- Python



Device Twin で機器を管理

デバイス



バックエンドのアプリ



テレメトリー

クラウドからメッセージを
デバイスに送る

クラウド側から指定可能な変数

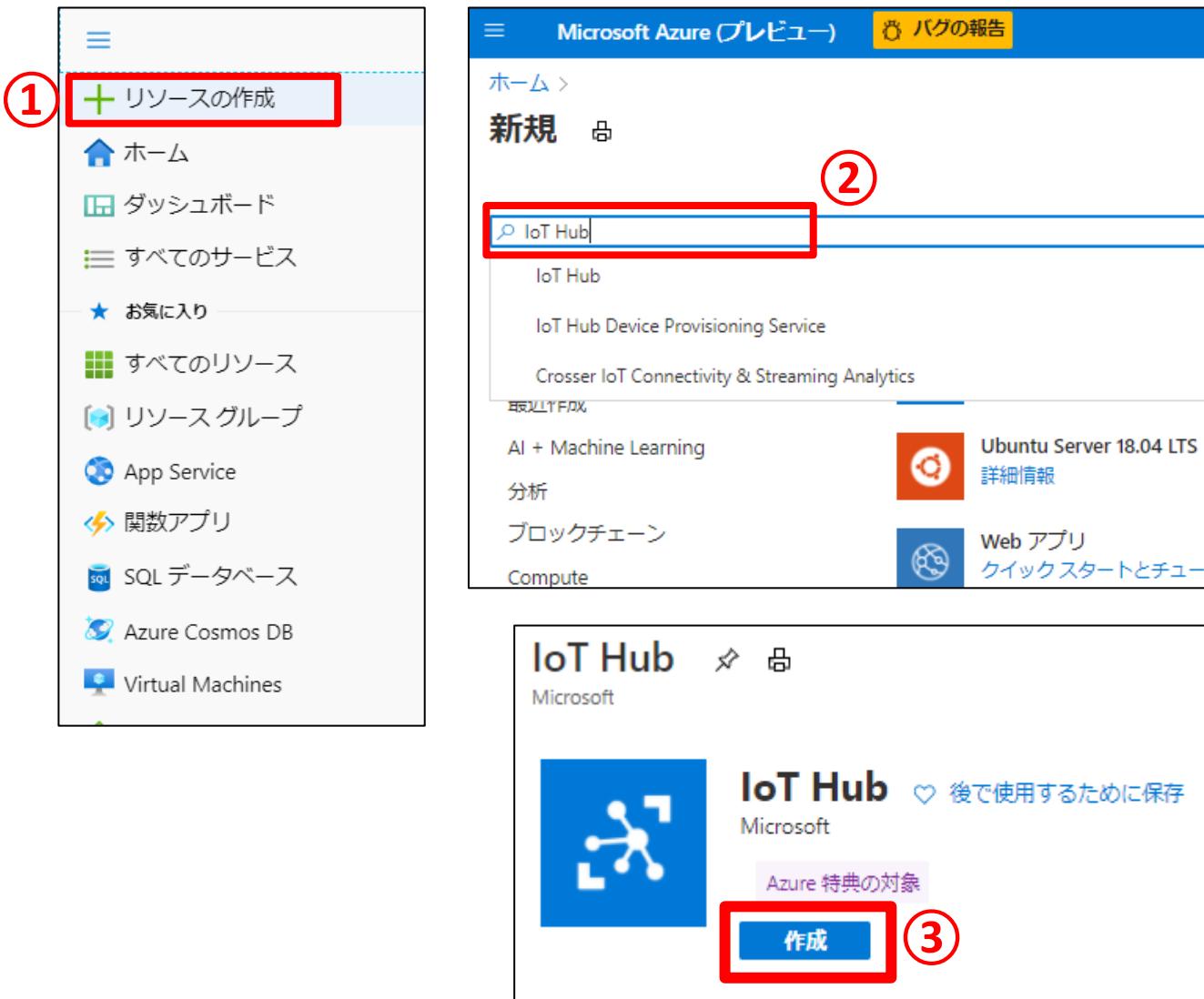
デバイス側由来の変数

クラウド側で付与可能な
メタデータ

経過も通知受信可能な
デバイスのメソッドを起動

クラウドのバックエンドサービス

IoT Hub の作成



1. Azure Portalへのアクセス

<https://ms.portal.azure.com>

2. 以下の順にメニューを押下

① [+ リソースの作成]

② [IoT Hub] を検索

③ [作成] を選択

IoT Hub の作成

IoT ハブ Microsoft

基本 ネットワーク 管理 タグ 確認および作成

何十億にも及ぶ IoT アセットに接続し、監視および管理できるように、IoT ハブを作成します。 詳細情報

プロジェクトの詳細

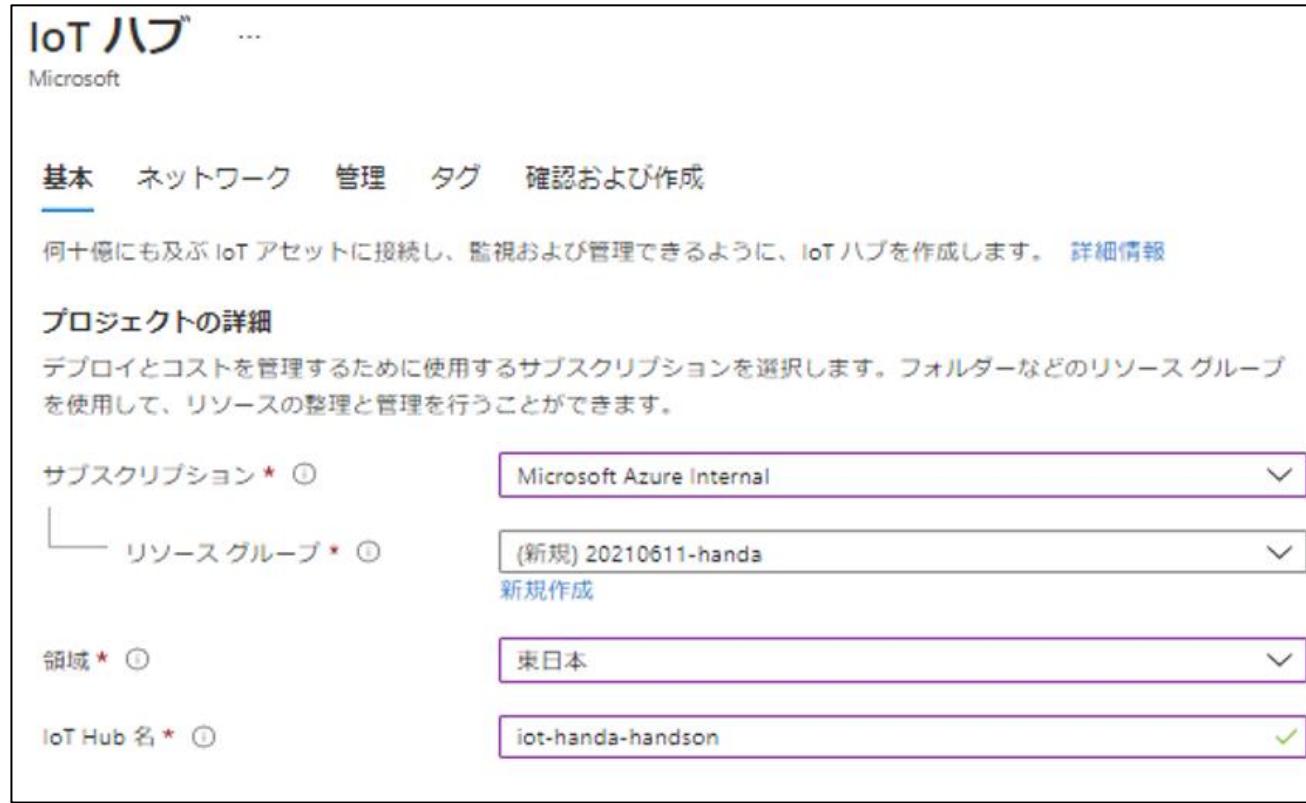
デプロイとコストを管理するために使用するサブスクリプションを選択します。フォルダーなどのリソース グループを使用して、リソースの整理と管理を行うことができます。

サブスクリプション * ① Microsoft Azure Internal

リソース グループ * ② (新規) 20210611-handa
新規作成

領域 * ③ 東日本

IoT Hub 名 * ④ iot-handa-handson



以下の項目を適宜設定し、[確認および作成]を押下します。

- ① サブスクリプション : (適切なものを選択ください)
- ② リソースグループ(※1) : (例) **YYYYMMDD-<name>**
- ③ リージョン : (例) 東日本
- ④ IoT Hub 名 : (例) **iot-<name>-handson**

(※1) リソースグループ : 本日のハンズオン用にご準備のある方は、既存リソースグループをご利用ください。
その他、新しく作成の場合は、新規作成を選択し、登録ください。

デプロイ前の確認画面

IoT ハブ Microsoft

検証に成功しました。

基本 ネットワーク 管理 タグ 確認および作成

①

サブスクリプション	Microsoft Azure Internal
リソース グループ	20210611-handa
領域	東日本
IoT Hub 名	iot-handa-handson

②

ネットワーク

接続方法	パブリック エンドポイント (すべてのネットワーク)
プライベート エンドポイント接続	なし

管理

価格とスケールティア	S1
IoT ハブ S1 のユニット数	1
1 日あたりのメッセージ	400,000
Device-to-cloud パーティション	4
月あたりのコスト	2800.00 JPY
Defender for IoT	参照: Defender for IoT の価格

タグ ②

作成 < 前へ: タグ 次へ > Automation オプション

設定内容に問題なければ、以下のステップを実施します。

① 設定内容の確認

こちらに先ほど設定した内容が表示されます

② [作成] ボタンにてデプロイ開始

※通常、1分～5分程度で完了いたします

デプロイ後、IoT デバイスの追加

Microsoft Azure (プレビュー) バグの報告 リソース、サービス、ドキュメントの検索 (G+/)

ホーム > iot-handa-handson

iot-handa-handson | IoT デバイス

検索 (Ctrl+ /) ② + 新規 最新の情報に更新 削除

概要 アクティビティ ログ アクセス制御 (IAM) タグ 問題の診断と解決 イベント 設定 共有アクセス ポリシー ID 価格とスケール ネットワーク 証明書 組み込みのエンドポイント フェールオーバー プロパティ ロック エクスプローラー クエリ エクスプローラー IoT デバイス IoT Edge

IoT Hub 内のデバイスを表示、作成、削除、更新します。

デバイスのクエリ

デバイス ID	状態	前回の状態の更新 (UTC)	認証の種類	クラウドからデバイスへのメッセージの数
デバイスが見つかりませんでした				

クエリ エディターに切り替える

Microsoft Azure (プレビュー) リソース、サービス

ホーム > iot-handa-handson > デバイスの作成

デバイス ID * ① device001

認証の種類 ① 対称キー X.509 自己署名済み X.509 CA 署名済み

主キー ① プライマリ キーを入力してください

セカンダリ キー ① セカンダリ キーを入力してください

自動生成キー ①

このデバイスを IoT ハブに接続する ① 有効化 無効化

親デバイス ① 親デバイスがありません
親デバイスの設定

保存

IoT Hubの画面にて、以下の順でボタンを押下します

① エクスプローラーにて[IoT デバイス]を選択

② [+新規]を選択

以下の項目を適宜設定ください

デバイスID：(例) device001

その他はデフォルト値のまま[保存]

接続文字列の取得

The screenshot shows the Microsoft Azure IoT Hub Device Management interface. On the left, a sidebar lists various management options like Overview, Metrics, Logs, and Device Explorer. The main area displays a list of devices, with 'device001' selected. A large arrow points from the device list to the detailed view on the right. In the detailed view, under the 'device001' card, the 'Primary Connection String' section is highlighted, showing the string value and a copy icon.

① 作成したIoTデバイスのデバイスIDをクリック

② [プライマリ接続文字列] をクリップボードにコピーします

2. サンプルプログラムの実行

接続文字列の設定

サンプルプログラム（SimulatedDevice.py）を開き、{IOTHUB_DEVICE_CONNECTION_STRING} を IoT Hubの画面より確認した接続文字列で置き換えます。

<変更前>

```
CONNECTION_STRING = "{IOTHUB_DEVICE_CONNECTION_STRING}"
```

<変更後> (例)

```
CONNECTION_STRING = "HostName=iot-handa-hanson.azure-devices.net;DeviceId=device001;SharedAccessKey=xxxxxxxxxxxxxxxxxxxxx=";
```

サンプルプログラムのダウンロード先

<https://github.com/yahanda/hands-on-iot-basics>

サンプルプログラムの内容

以下項目を含むJSONを、継続的に IoT Hub に送信する内容としています

- Start_time 作業開始時刻
- End_time 作業終了時刻
- Work 作業工程 (WORK01~WORK12まで)
- Line_num ライン番号 (LINE001固定)
- Lot_num ロット番号 (WORK01~WORK12までの1サイクルを1ロットとして、LOT001からインクリメント)
- Serial_num シリアル番号 (1から順にインクリメント)

サンプルプログラムで送信するJSON例

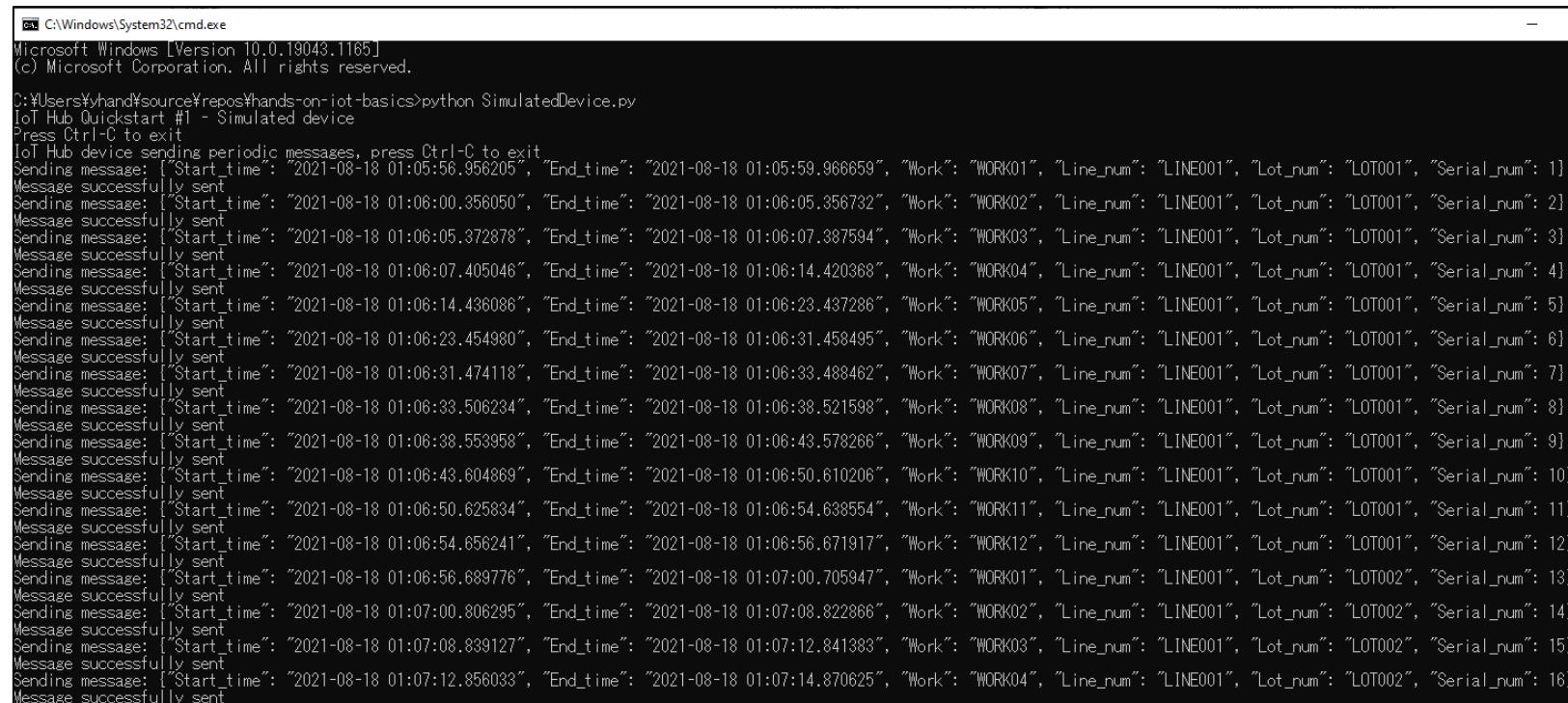
```
{"Start_time": "2021-08-18 01:05:56.956205", "End_time": "2021-08-18 01:05:59.966659", "Work": "WORK01", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 1}  
{"Start_time": "2021-08-18 01:06:00.356050", "End_time": "2021-08-18 01:06:05.356732", "Work": "WORK02", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 2}  
{"Start_time": "2021-08-18 01:06:05.372878", "End_time": "2021-08-18 01:06:07.387594", "Work": "WORK03", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 3}  
:  
{"Start_time": "2021-08-18 01:06:54.656241", "End_time": "2021-08-18 01:06:56.671917", "Work": "WORK12", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 12}  
{"Start_time": "2021-08-18 01:06:56.689776", "End_time": "2021-08-18 01:07:00.705947", "Work": "WORK01", "Line_num": "LINE001", "Lot_num": "LOT002", "Serial_num": 13}  
{"Start_time": "2021-08-18 01:07:00.806295", "End_time": "2021-08-18 01:07:08.822866", "Work": "WORK02", "Line_num": "LINE001", "Lot_num": "LOT002", "Serial_num": 14}  
:  
{"Start_time": "2021-08-18 01:07:38.174552", "End_time": "2021-08-18 01:07:41.188959", "Work": "WORK12", "Line_num": "LINE001", "Lot_num": "LOT002", "Serial_num": 24}  
{"Start_time": "2021-08-18 01:07:41.205187", "End_time": "2021-08-18 01:07:42.221767", "Work": "WORK01", "Line_num": "LINE001", "Lot_num": "LOT003", "Serial_num": 25}  
{"Start_time": "2021-08-18 01:07:42.239488", "End_time": "2021-08-18 01:07:45.251910", "Work": "WORK02", "Line_num": "LINE001", "Lot_num": "LOT003", "Serial_num": 26}  
:
```

サンプルプログラムの実行

コマンドプロンプトを起動し、SimulatedDevice.py が配置されたフォルダで、以下コマンドを実行します。

>python SimulatedDevice.py

IoT Hub に送信するJSONメッセージがコンソール上に表示されます。



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vhandy\source\repos\hands-on-iot-basics>python SimulatedDevice.py
IoT Hub Quickstart #1 - Simulated device
Press Ctrl-C to exit
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"Start_time": "2021-08-18 01:05:56.956205", "End_time": "2021-08-18 01:05:59.966659", "Work": "WORK01", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 1}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:00.356050", "End_time": "2021-08-18 01:06:05.356732", "Work": "WORK02", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 2}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:05.372878", "End_time": "2021-08-18 01:06:07.387594", "Work": "WORK03", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 3}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:07.405046", "End_time": "2021-08-18 01:06:14.420368", "Work": "WORK04", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 4}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:14.436086", "End_time": "2021-08-18 01:06:23.437286", "Work": "WORK05", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 5}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:23.454980", "End_time": "2021-08-18 01:06:31.458495", "Work": "WORK06", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 6}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:31.474118", "End_time": "2021-08-18 01:06:33.488462", "Work": "WORK07", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 7}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:33.506234", "End_time": "2021-08-18 01:06:38.521598", "Work": "WORK08", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 8}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:38.553958", "End_time": "2021-08-18 01:06:43.578266", "Work": "WORK09", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 9}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:43.604869", "End_time": "2021-08-18 01:06:50.610206", "Work": "WORK10", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 10}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:50.625834", "End_time": "2021-08-18 01:06:54.638554", "Work": "WORK11", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 11}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:54.656241", "End_time": "2021-08-18 01:06:56.671917", "Work": "WORK12", "Line_num": "LINE001", "Lot_num": "LOT001", "Serial_num": 12}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:06:56.689776", "End_time": "2021-08-18 01:07:00.705947", "Work": "WORK01", "Line_num": "LINE001", "Lot_num": "LOT002", "Serial_num": 13}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:07:00.806295", "End_time": "2021-08-18 01:07:08.822866", "Work": "WORK02", "Line_num": "LINE001", "Lot_num": "LOT002", "Serial_num": 14}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:07:08.839127", "End_time": "2021-08-18 01:07:12.841383", "Work": "WORK03", "Line_num": "LINE001", "Lot_num": "LOT002", "Serial_num": 15}
Message successfully sent
Sending message: {"Start_time": "2021-08-18 01:07:12.856033", "End_time": "2021-08-18 01:07:14.870625", "Work": "WORK04", "Line_num": "LINE001", "Lot_num": "LOT002", "Serial_num": 16}
Message successfully sent
```

IoT Hub の監視

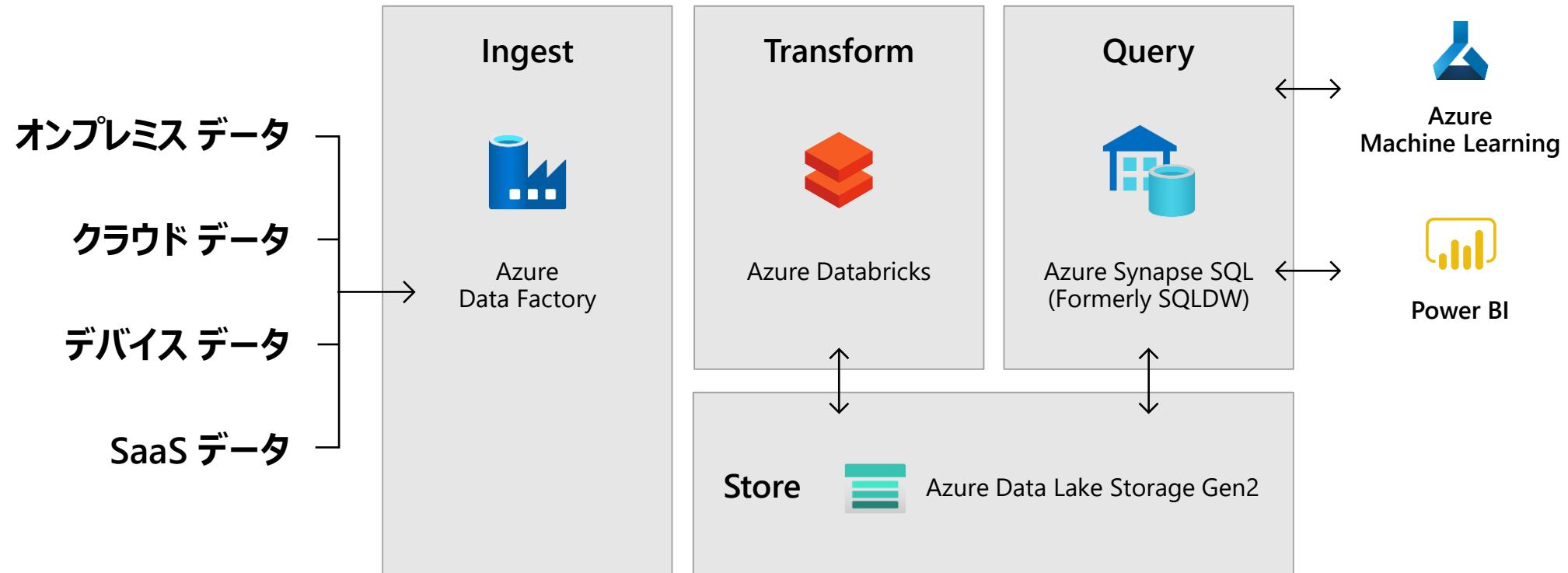
IoT Hub の画面の[概要] を開き、正しくメッセージが IoT Hub に送られていることを確認します。



※メッセージ数の値の表示は1分程度遅れがあります

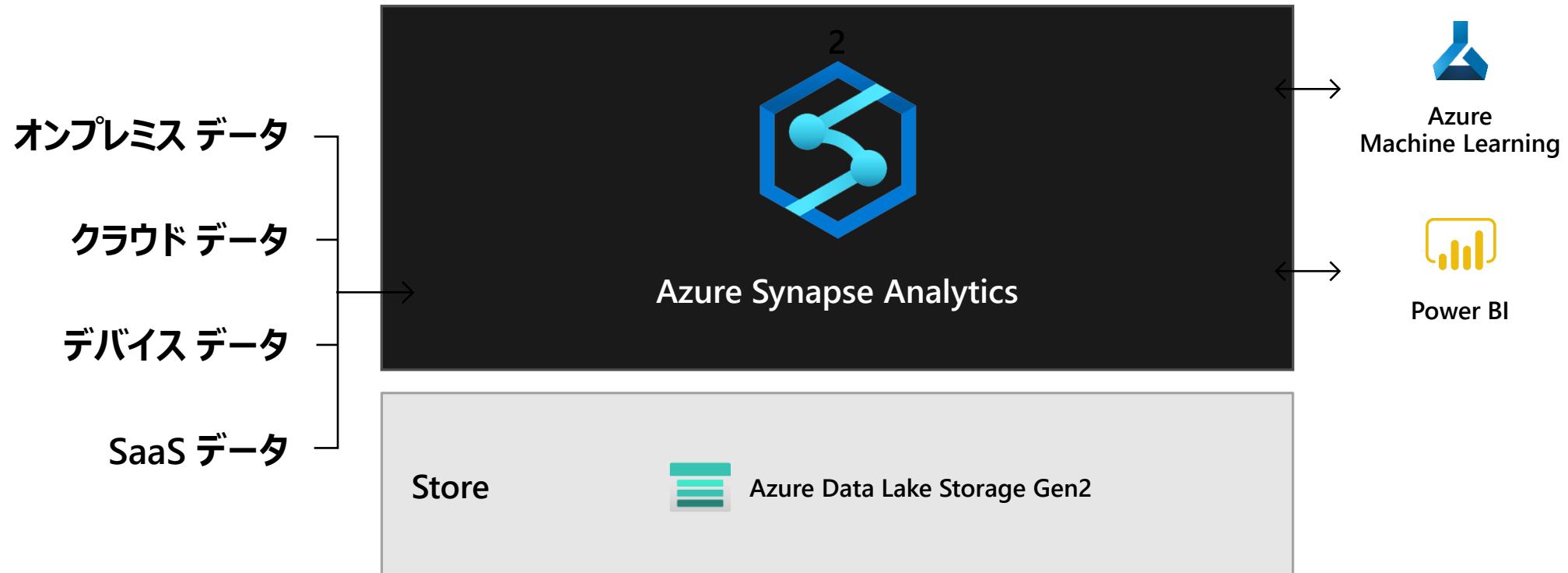
3. Synapse Analytics の作成

新しい Azure Synapse Analytics としてのイノベーション



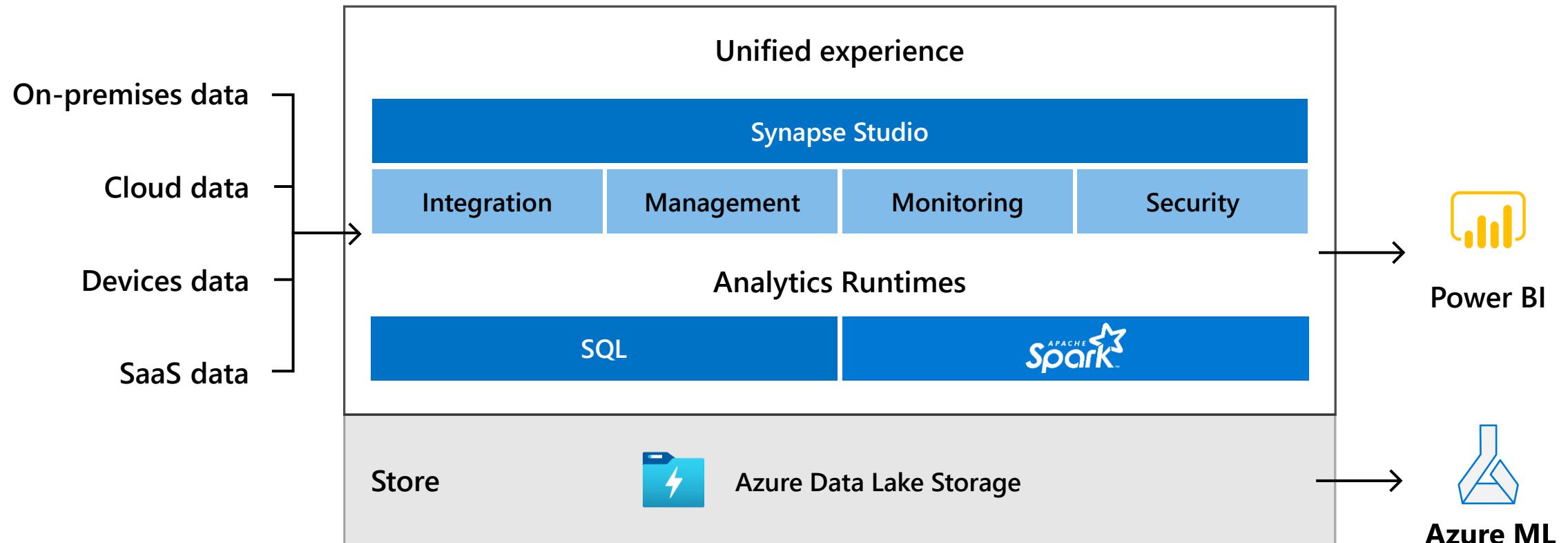
新しい Azure Synapse Analytics としてのイノベーション

データ ウェアハウスとビッグデータ分析システムを融合させ、
あらゆるデータから、驚異的なスピードでインサイトを提供する
無制限の分析サービス

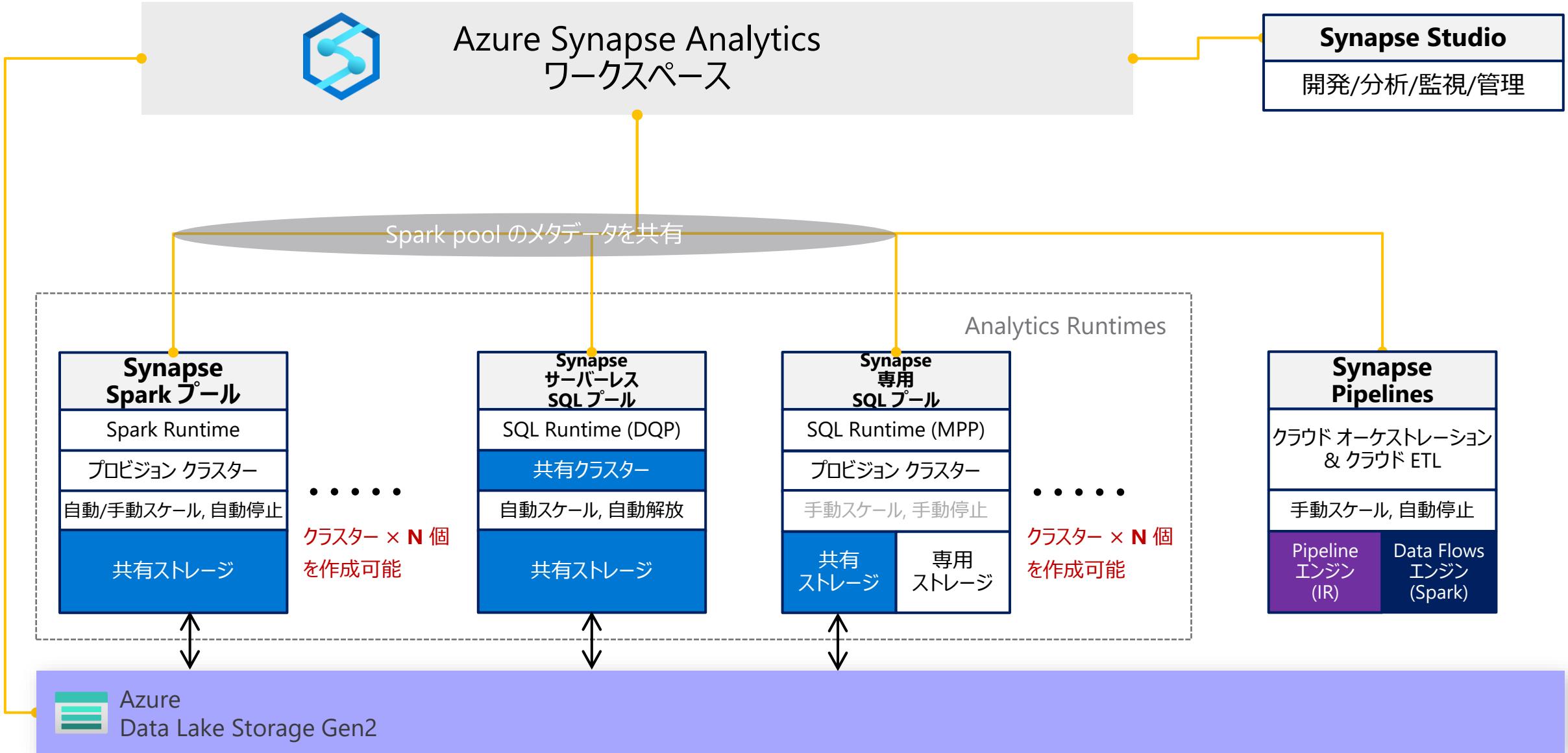


Azure Synapse Analytics のアーキテクチャ

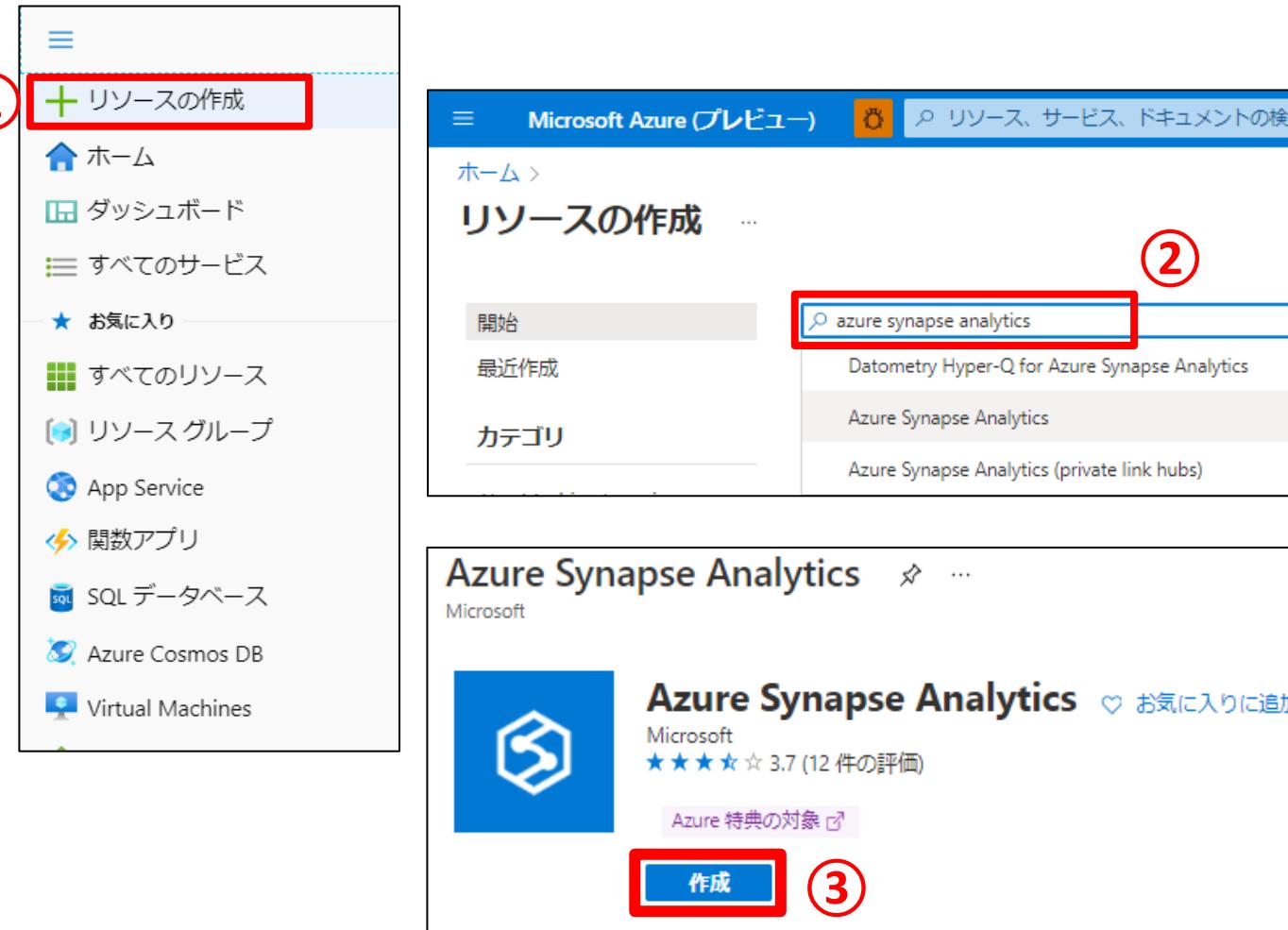
Synapse Analytics のアーキテクチャ



Azure Synapse Analytics の物理アーキテクチャ



Synapse Analytics の作成



1. Azure Portalへのアクセス

<https://ms.portal.azure.com>

2. 以下の順にメニューを押下

① [+ リソースの作成]

② [Azure Synapse Analytics] を検索

③ [作成] を選択

Synapse Analytics の各種設定

Synapse ワークスペースの作成 ...

*基本 *セキュリティ ネットワーク タグ レビュー + 作成

数回クリックするだけで、エンタープライズ分析ソリューションを開発するための Synapse ワークスペースを作成できます。

プロジェクトの詳細

デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソースグループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション * ① Microsoft Azure Internal (1)

リソースグループ * ② 20210611-handa (2)
新規作成

マネージド リソース グループ ③ 管理対象リソース グループ名を入力してください

ワークスペースの詳細

ワークスペースに名前を付け、場所を選択し、ログとジョブの出力の既定の場所として機能するプライマリ Data Lake Storage Gen2 ファイルシステムを選択します。

ワークスペース名 * ④ syn-handa-handson (3)

地域 * ⑤ 東日本 (4)

Data Lake Storage Gen2 を選択する * ⑥ (新規) dlshandahandson (5)
URL を使用して手動で
新規作成

アカウント名 * ⑦ (新規) users (6)
新規作成

ワークスペースでストレージ BLOB のクエリを対話形式で実行するには、
Data Lake Storage Gen2 アカウントでそのストレージ BLOB のデータ共同作成者ロールを自分に割り当てます。

確認および作成 < 前へ 次へ: セキュリティ >

以下の項目を、適宜設定ください。

※必要に応じて、設定内容をメモお願いします

- ① サブスクリプション : (例) **syn-<name>-handson**
- ② リソースグループ :
- ③ ワークスペース名 : (例) **dls<name>handson**
- ④ 場所 :
- ⑤ アカウント名 : (例) **users**
- ⑥ ファイルシステム名 : (例) **users**

[次へ: セキュリティ >] ボタンを押下

Synapse Analytics の各種設定

Synapse ワークスペースの作成 ...

* 基本 *セキュリティ ネットワーク タグ レビュー + 作成

ワークスペースのセキュリティ オプションを構成します。

SQL 管理者の資格情報

ワークスペースの SQL プールに対する管理者アクセスに使用できる資格情報を指定してください。パスワードを指定しない場合は、自動的に生成されます。パスワードは後で変更できます。

管理者ユーザー名* ①

パスワード ② ✓

パスワードの確認 ✓

ワークスペースの暗号化

⚠️ ワークスペースの作成時にカスタマー マネージド キーの使用を選択すると、二重暗号化構成を変更できなくなります。

自分が管理するキー (カスタマー マネージド キー) を使用して、ワークスペース内のすべての保存データを暗号化することを選択します。これにより、プラットフォーム マネージド キーを使用するインフラストラクチャ レイヤーでの暗号化を使用した二重暗号化が提供されます。 [詳細情報](#)

カスタマー マネージド キーを使用する二重暗号化を有効にする

システム割り当てマネージド ID

ワークスペースのシステム割り当て ID に割り当てるアクセス許可を選択します。 [詳細情報](#)

パイプライン (ワークスペースのシステム割り当て ID として実行中) による SQL プールへのアクセスを許可します。 ①

Data Lake Storage Gen2 アカウントへのネットワーク アクセスを許可します。 ①

① 選択した Data Lake Storage Gen2 アカウントでは、ネットワーク アクセス ルールを使用したネットワーク アクセスが制限されません。または、[基本] タブの下にある URL を使用して手動でストレージアカウントを選択しました。 [詳細](#)

確認および作成 < 前へ 次へ: ネットワーク >

以下の項目を、適宜設定ください。

※必要に応じて、設定内容をメモお願いします

- ① 管理者ユーザー名 : (例) **sqladminuser**
② パスワード :

その他はデフォルト値のまま、[確認および作成] ボタンを押下

デプロイ前の確認画面

Synapse ワークスペースの作成 ...

✓ 検証に成功しました

* 基本 *セキュリティ ネットワーク タグ レビュー + 作成

製品の詳細

Azure Synapse Analytics ワークスペース サーバーレス SQL 推定コスト/TB ①
Microsoft 提供 560.00 JPY

使用条件 | プライバシー ポリシー

①

使用条件

[作成] をクリックすることで、お客様は (a) 上記の Marketplace のオファーに関する法律条項とプライバシーに関する声明に同意し、(b) Microsoft がそのオファーに関する料金を現在の支払い方法で Azure サブスクリプションと同じ請求頻度で請求することを認め、かつ、(c) Microsoft がお客様の連絡先情報、使用量情報、取引に関する情報を、サポート、請求、その他の取引上のアクティビティを目的として、オファーのプロバイダーと共有する可能性があることに同意します。 Microsoft は、サードパーティのオファーに対する権利は提供しません。その他の詳細については、以下を参照してください [Azure Marketplace 使用条件](#) ②

基本

サブスクリプション	Microsoft Azure Internal
リソース グループ	20210611-handa
地域	東日本
ワークスペース名	(新規) syn-handa-handson
Data Lake Storage Gen2 アカウント	(新規) https://dlshandahandson.dfs.core.windows.net
Data Lake Storage Gen2 ファイルシステム	(新規) users
マネージド リソース グループ	なし
ロールの割り当て	ストレージ BLOB データ共同作成者ロールは、指定された Data Lake Storage Gen2 アカウントのワークスペース マネージド ID と現在のユーザーの両方に割り当てられます。

セキュリティ

作成 ② < 前へ 次へ > Automation のテンプレートをダウンロードする

設定内容に問題なければ、以下のステップを実施します。

- ① 設定内容の確認
こちらに先ほど設定した内容が表示されます
- ② [作成] ボタンにてデプロイ開始
※通常、5分～10分程度で完了いたします

4. Stream Analytics の作成

Stream Analytics : 概要

Azure 上でのストリーム データのリアルタイム処理を行うサービス

- ・ デバイス、マシーン、アプリケーションと接続した Event Hubs から 数百万のリアル タイム イベントを取得して解析
- ・ リアルタイム分析ソリューションを低コストで実装し、保守できるように最適化
- ・ 一連のタスク（入力 → クエリ → 出力）をジョブといい、開始・停止が可能



Azure Stream Analytics のコード削減効果

コードの削減 = 開発者の生産性向上

オープンソースでは 1,915 行のコードが

```
@ApplicationAnnotation(name="WordCountDemo")
public class Application implements StreamingApplication
{
    protected String fileName =
        "com/datatorrent/demos/wordcount/samplefile.txt";
        private Locality locality = null;

    @Override public void populateDAG(DAG dag, Configuration
conf)
    {
        locality = Locality.CONTAINER_LOCAL;
        WordCountInputOperator input =
dag.addOperator("wordinput", new
WordCountInputOperator());
        input.setFileName(fileName);
        UniqueCounter<String> wordCount =
dag.addOperator("count", new
{
    .....
}
```

Stream Analytics ではわずか 3 行

```
SELECT Avg(Purchase), ScoreTollId, Count(*)
FROM GameDataStream
GROUP BY TumblingWindows(5, Minute), Score
```

データ操作

```
SELECT
FROM
WHERE
HAVING
GROUP BY
CASE WHEN THEN
ELSE
INNER/LEFT OUTER
JOIN
UNION
CROSS/OUTER
APPLY
CAST INTO
ORDER BY ASC, DSC
```

集計

```
SUM
COUNT
AVG
MIN
MAX
STDEV
STDEVP
VAR
VARP
TopOne
```

日付/時刻関数

```
DateName
DatePart Day, Month, Year
DateDiff
DateTimeFromParts
DateAdd
```

テンポラル

```
Lag
IsFirst
Last
CollectTop
```

Window 拡張関数

```
TumblingWindow
HoppingWindow
SlidingWindow
```

スケーリング拡張関数

```
WITH
PARTITION BY
OVER
```

文字列関数

```
Len
Concat
CharIndex
Substring
Lower, Upper
PatIndex
```

数学関数

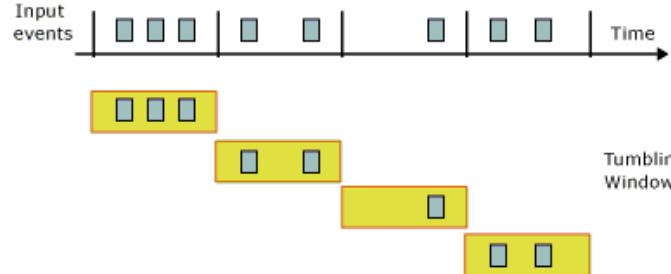
```
ABS
CEILING
EXP
FLOOR
POWER
SIGN
SQUARE
SQRT
```

地理空間関数 (プレビュー)

```
CreatePoint
CreatePolygon
CreateLineString
ST_DISTANCE
ST_WITHIN
ST_OVERLAPS
ST_INTERSECTS
```

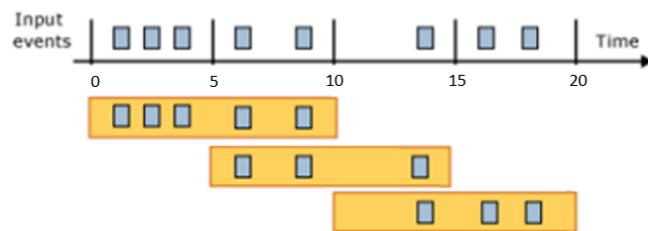
重要な3つのWindowing

- Tumbling Window



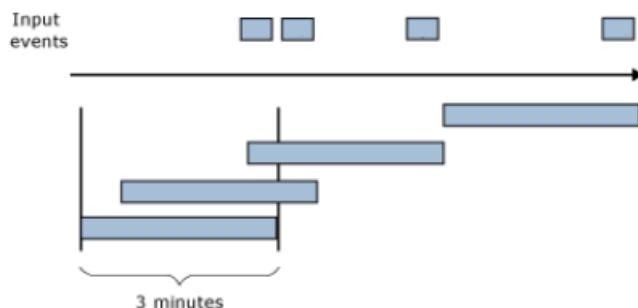
```
SELECT System.TimeStamp AS OutTime, TollId, COUNT (*)  
FROM Input TIMESTAMP BY EntryTime  
GROUP BY TollId, TumblingWindow(Duration(minute,5))
```

- Hopping Window



```
SELECT System.TimeStamp AS OutTime, TollId, COUNT (*)  
FROM Input TIMESTAMP BY EntryTime  
GROUP BY TollId, HoppingWindow(Duration(minute, 10),  
Hop(minute, 5))
```

- Sliding Window

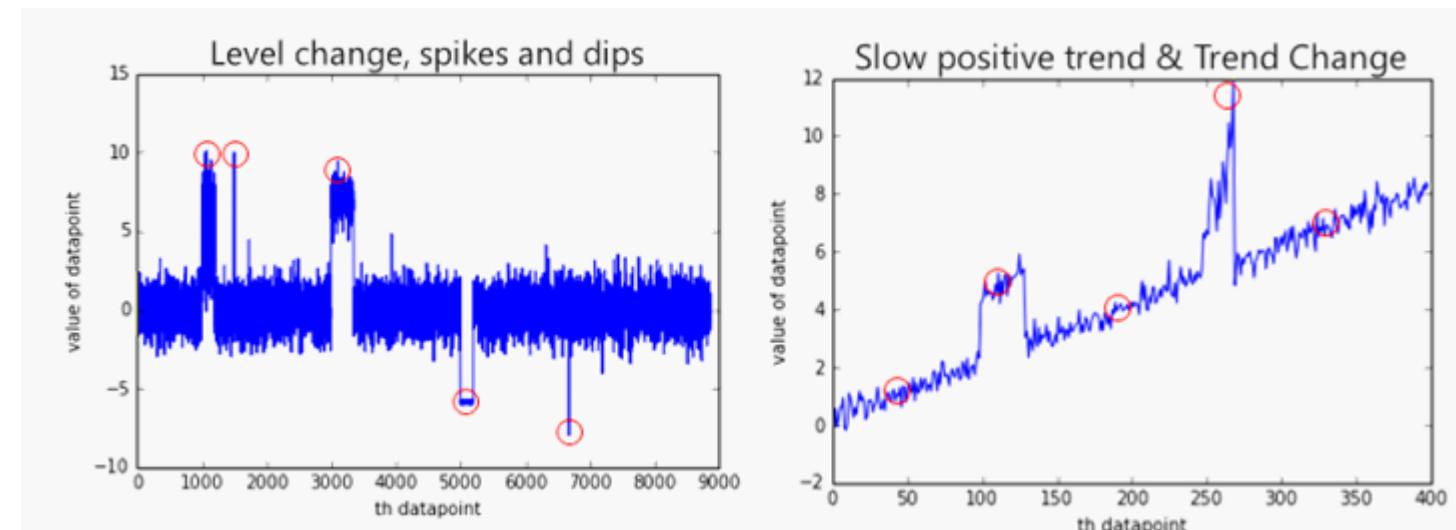


```
SELECT System.TimeStamp AS OutTime, TollId, COUNT (*)  
FROM Input TIMESTAMP BY EntryTime  
GROUP BY TollId, SlidingWindow(Duration(minute, 3))  
HAVING Count(*) > 3
```

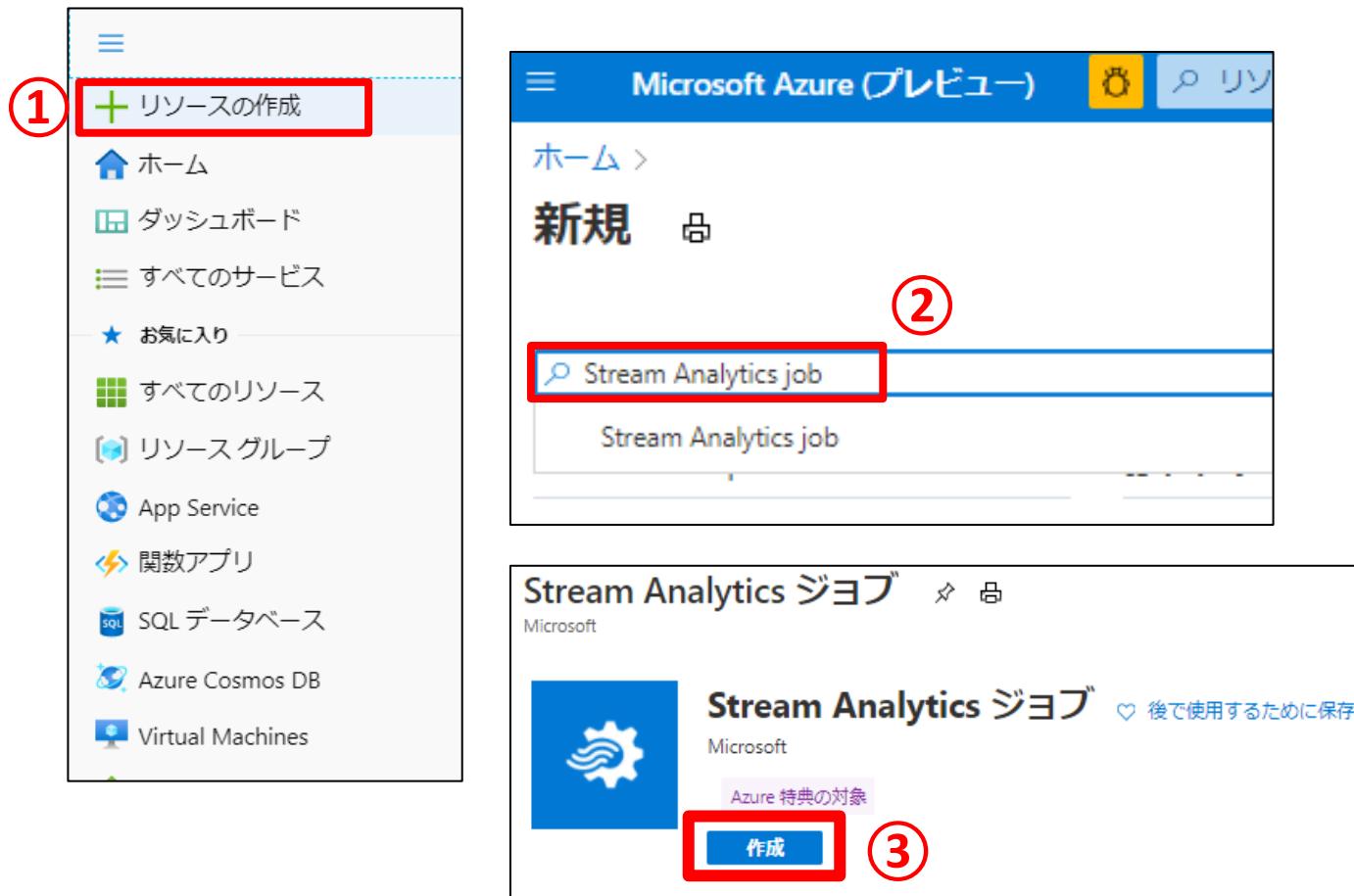
Azure Stream Analytics を使った Edge での異常検知

- ✓ クラウド・エッジ側で、タイムシリーズデータのリアルタイム異常検知が可能
- ✓ クエリ言語を使って、簡単に開発が可能
- ✓ 5種類の異常検知に対応
 - ✓ Spikes, Dips, Slow positive/negative trend, Bi-level change

```
OVER  
[WHEN  
    boolean_expression])  
  
AnomalyDetection_SpikeAndDip(  
    <scalar_expression>,  
    <confidence>,  
    <historySize>)  
  
([PARTITION BY <partition key>]  
LIMIT DURATION(<unit>,<length>)
```



Stream Analytics の作成



1. Azure Portalへのアクセス

<https://ms.portal.azure.com>

2. 以下の順にメニューを押下

① [+ リソースの作成]

② [Stream Analytics job] を検索

③ [作成] を選択

Stream Analytics の各種設定

新しい Stream Analytics ジョブ ...

これにより、新しい Stream Analytics ジョブが作成されます。Azure Stream Analytics は、データを即時に分析して、その結果を Azure データベースや Azure Storage などに送信するためのサービスです。

ジョブ名 *

asa-handa-handson ✓ (1)

サブスクリプション *

Microsoft Azure Internal (2)

リソース グループ *

20210611-handa (3)

新規作成

場所 *

東日本 (4)

ホスティング環境 (1)

クラウド Edge

ストリーミング ユニット (1 から 192) (1)

3

このジョブに必要なすべてのプライベートデータ資産を自分のストレージ アカウントに保管してセキュリティで保護する。 (1)

作成

以下の項目を、適宜設定ください。

※必要に応じて、設定内容をメモお願いします

(例) **asa-<name>-handson**

(1) ジョブ名 :

(2) サブスクリプション :

(3) リソース グループ :

(4) 場所 :

その他はデフォルト値のまま、[作成] にてデプロイ開始

Stream Analytics の入力設定

Stream Analyticsの画面にて、以下の順でボタンを押下

- ① ジョブトポロジにて[入力]を選択
- ② [ストリーム入力の追加] → [IoT Hub] を選択

asa-handa-handson | 入力

Stream Analytics ジョブ

検索 (Ctrl+ /)

概要

アクティビティ ログ

アクセス制御 (IAM)

タグ

問題の診断と解決

設定

プロパティ

ロック

ジョブ トポロジ

① 入力

関数

クエリ

出力

IoT Hub

新規入力

入力のエイリアス *

iothub

IoT Hub 設定を手動で行う

サブスクリプションから IoT Hub を選択する

サブスクリプション

Microsoft Azure Internal

IoT Hub *

iot-handa-handson

コンシューマー グループ *

\$Default

共有アクセス ポリシー名 *

iothubowner

共有アクセス ポリシー キー

エンドポイント

メッセージ

パーティション キー

イベントシリアル化形式 *

JSON

エンコード

UTF-8

イベントの圧縮タイプ

なし

保存

選択したリソースと Stream

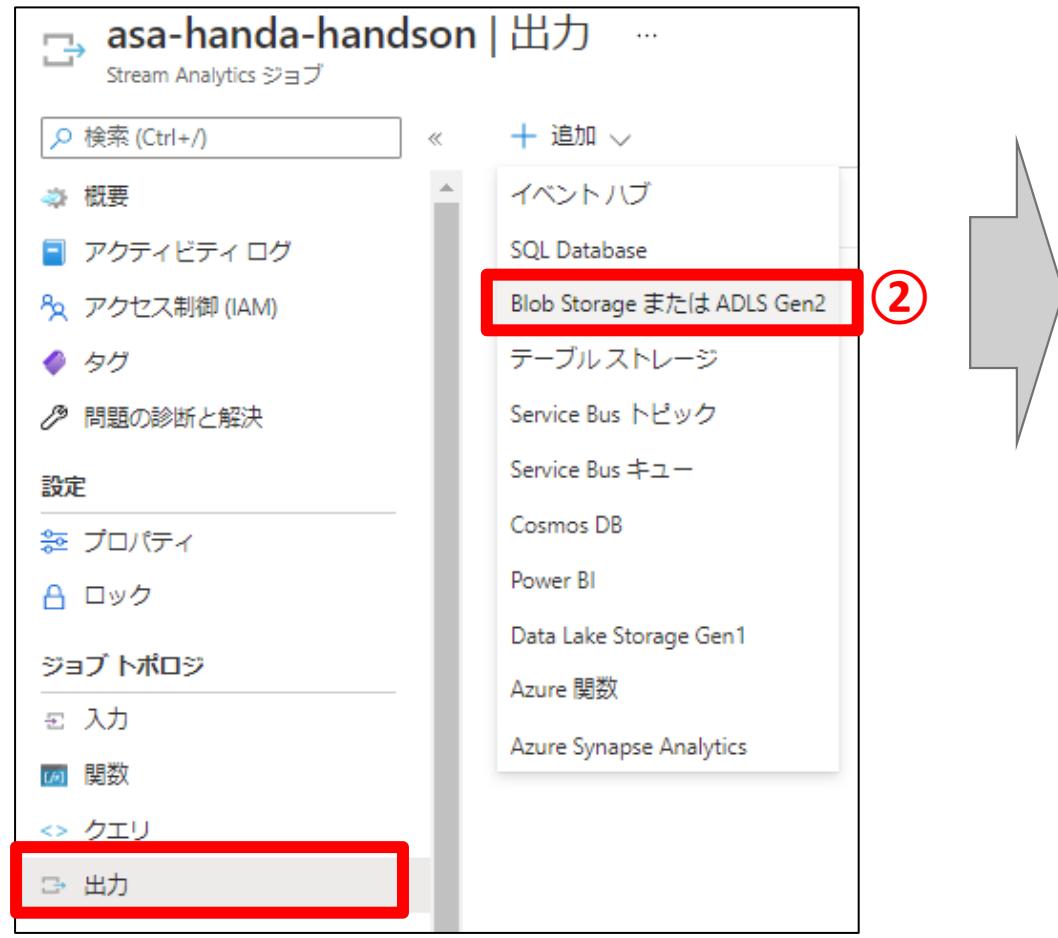
以下の項目を適宜設定し[保存]をクリック

- ✓ 入力のエイリアス : (例) **iothub**
- ✓ IoT Hub : 以前の手順で作成したIoT Hubを選択
- ✓ その他はデフォルト設定

Stream Analytics の出力設定 (Storage)

Stream Analyticsの画面にて、以下の順でボタンを押下

- ① ジョブトポロジにて[出力]を選択
- ② [追加] → [Blob Storage または ADLS Gen2] を選択



Blob Storage または ADLS Gen2 ×
新規出力

出力エイリアス*
adls

Blob Storage または ADLS Gen2設定を手動で行う
 サブスクリプションからBlob Storage または ADLS Gen2を選択する

サブスクリプション
Microsoft Azure Internal

ストレージアカウント*
dlshandahandson

コンテナー*
 新規作成 既存のものを使用
iotdata

認証モード
接続文字列

ストレージアカウントキー

パスパターン①
stream/device/{date}

日付の形式
YYYY/MM/DD

時刻の形式
HH

イベントシリアル化形式*①
PARQUET

最小行数①
2000

最大時間
時間① 分
0 1

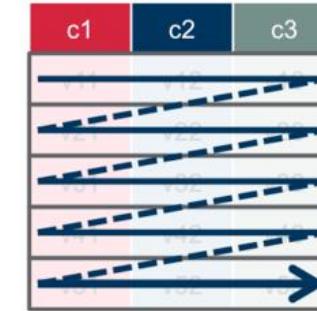
保存 選択したリソースと Stream

以下の項目を適宜設定し[保存]をクリック

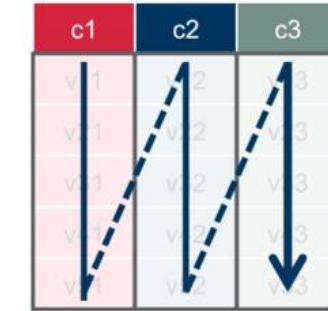
- ✓ 出力エイリアス : (例) **adls**
- ✓ ストレージアカウント : 以前の手順で作成したストレージアカウントを選択
- ✓ コンテナー : **新規作成**
- ✓ 名前 : (例) **iotdata**
- ✓ 認証モード : **接続文字列**
- ✓ パス パターン : **stream/device/{date}**
- ✓ イベントシリアル化形式 : **PARQUET**
- ✓ その他はデフォルト設定

参考：Parquet とは

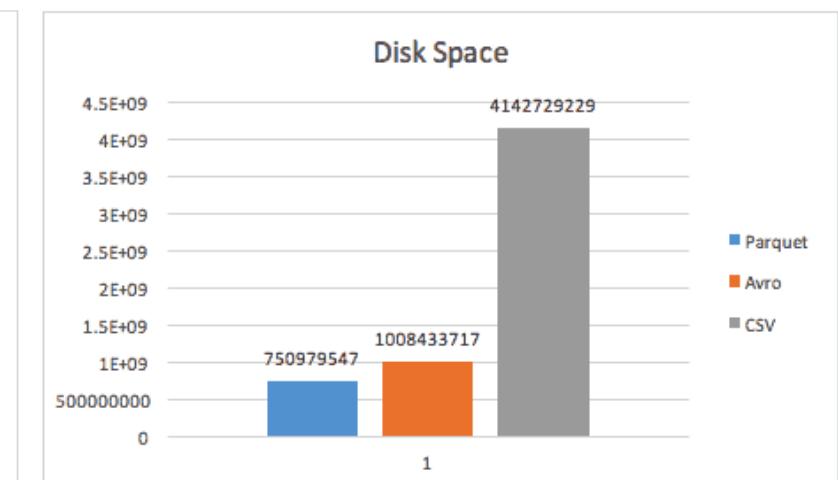
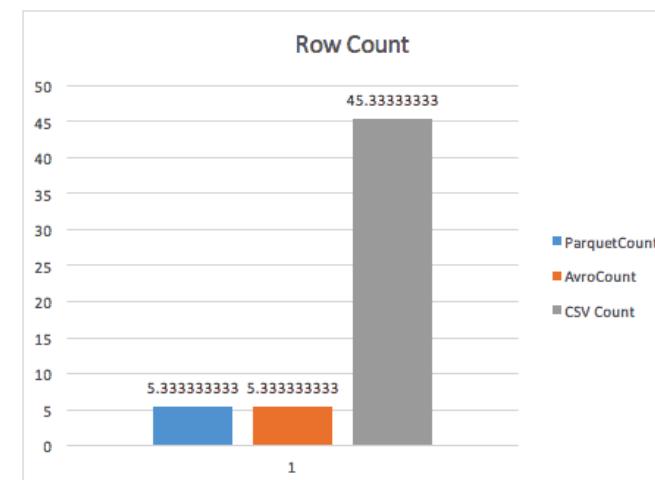
- 列指向データフォーマット
- CSVなどの行指向と比較して、圧縮効率や列に対する集計処理などにおいてアドバンテージを持つ
 - 必要なカラムのみを読み込むことでIOを削減し、データアクセス速度の向上を実現
 - ストレージコストの削減にも寄与
- 一般的に、IoTで扱うような分析用途のビッグデータ基盤に向いている



Row-oriented フォーマット
(CSV, TSV, ...)



Column-oriented フォーマット
(Parquet, ORC, ...)

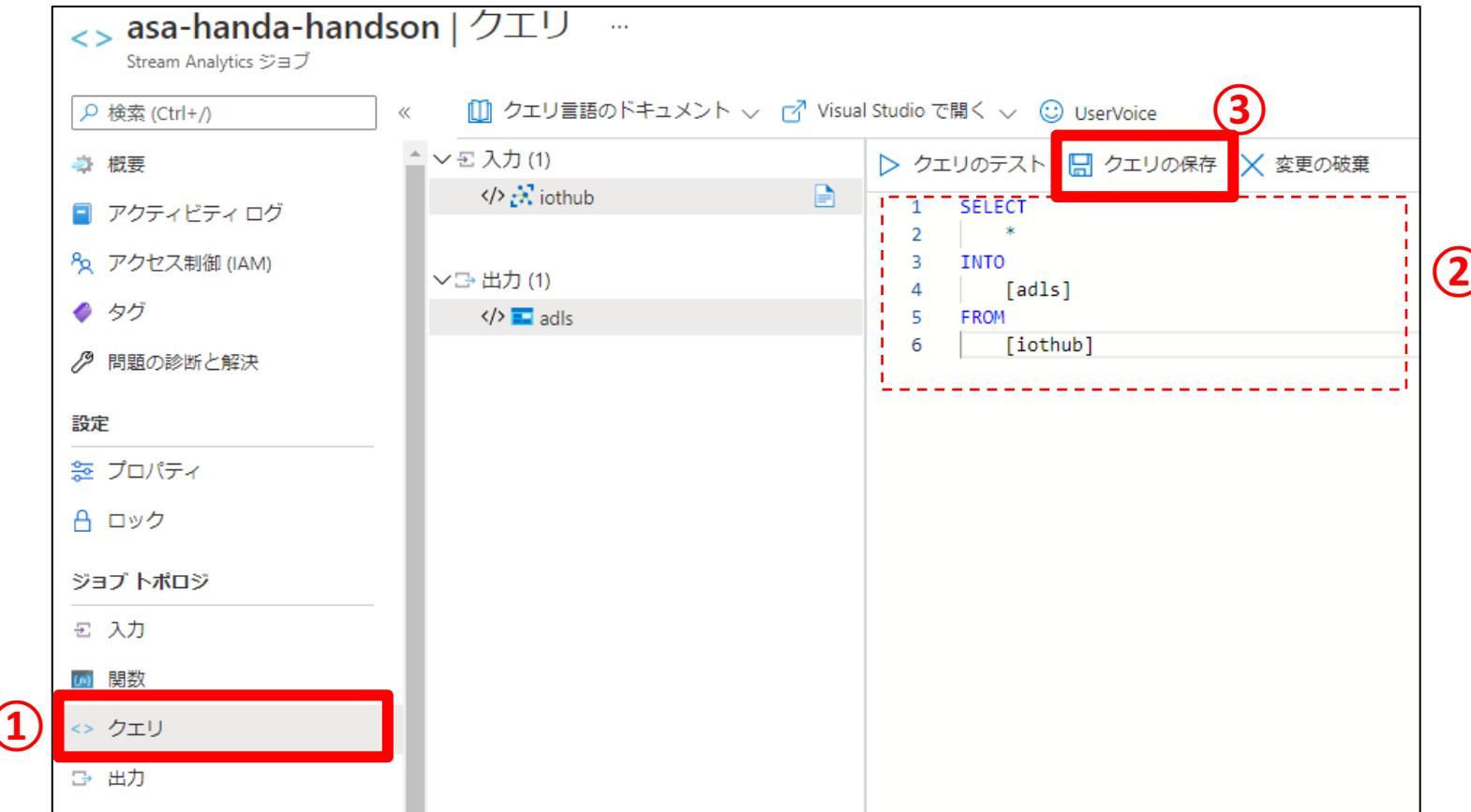


参考 : Benchmarking Apache Parquet: The Allstate Experience
<https://blog.cloudera.com/benchmarking-apache-parquet-the-allstate-experience/>

Stream Analytics のクエリ設定

Stream Analyticsの画面にて、以下の順でボタンを押下します。

- ① ジョブトポロジにて[クエリ]を選択



- ② 以下のクエリ内容を入力

```
SELECT
*
INTO
[adls]
FROM
[iothub]
```

- ③ [クエリの保存] をクリック

補足：クエリのテスト

[入力のプレビュー] タブで、IoT Hubからのイベントがプレビュー可能です

その状態で[クエリのテスト] をクリックすると、クエリの実行結果が[テスト結果] タブに表示されます

The screenshot shows the Azure IoT Explorer interface with the 'Query Test' tab selected (highlighted by a red box). The query code is:

```
1 SELECT
2     Start_time, End_time, Work
3 INTO
4     [adls]
5 FROM
6     [iothub]
```

The results are displayed in two tabs: 'Input Preview' (highlighted by a red box) and 'Test Result' (also highlighted by a red box). The 'Input Preview' tab shows the first 50 rows of data from the 'adls' table. The 'Test Result' tab shows the results of the query execution, which displays the same data as the input preview.

Input Preview (Top Table):

Start_time	End_time	Work
"2021-08-18 16:27:40.128983"	"2021-08-18 16:27:47.129468"	"WORK08"
"2021-08-18 16:27:30.102210"	"2021-08-18 16:27:40.102983"	"WORK07"
"2021-08-18 16:27:23.075560"	"2021-08-18 16:27:30.076218"	"WORK06"
"2021-08-18 16:27:18.046196"	"2021-08-18 16:27:23.046558"	"WORK05"
"2021-08-18 16:27:08.014190"	"2021-08-18 16:27:18.014236"	"WORK04"
"2021-08-18 16:27:06.988364"	"2021-08-18 16:27:07.989188"	"WORK03"
"2021-08-18 16:27:05.968295"	"2021-08-18 16:27:06.968377"	"WORK02"
"2021-08-18 16:27:02.936326"	"2021-08-18 16:27:05.937295"	"WORK01"
"2021-08-18 16:26:55.911136"	"2021-08-18 16:27:02.9111326"	"WORK12"
"2021-08-18 16:26:46.885031"	"2021-08-18 16:26:55.885137"	"WORK11"
"2021-08-18 16:26:44.858421"	"2021-08-18 16:26:46.858849"	"WORK10"

Test Result (Bottom Table):

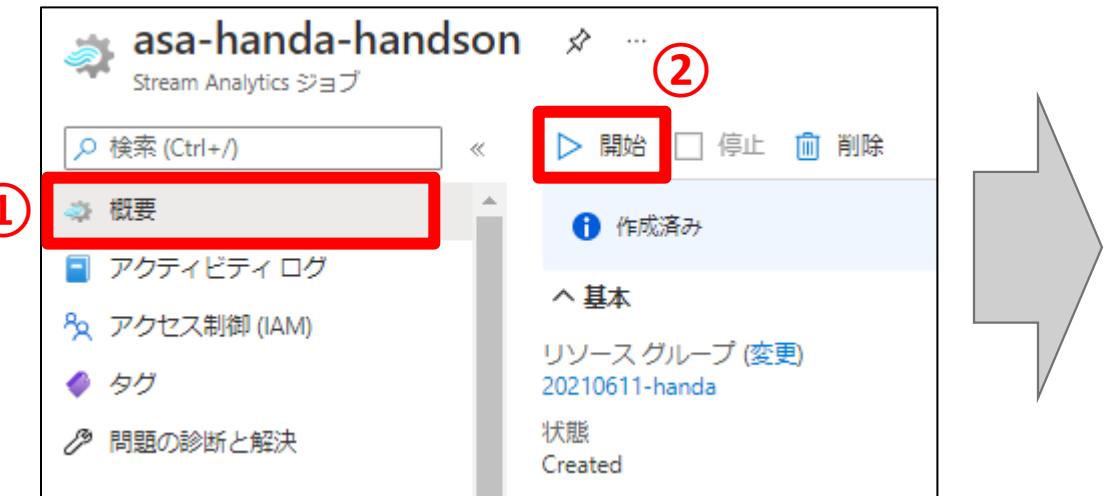
Start_time	End_time	Work	Line	Line	Line	Line	Line
"2021-08-18 16:27:40....	"2021-08-18 16:27:47....	"WORK08"	"LINE001"	"LOT132"	1580	"2021-08-18T08:23:19....	0
"2021-08-18 16:27:30....	"2021-08-18 16:27:40....	"WORK07"	"LINE001"	"LOT132"	1579	"2021-08-18T08:23:19....	0
"2021-08-18 16:27:23....	"2021-08-18 16:27:30....	"WORK06"	"LINE001"	"LOT132"	1578	"2021-08-18T08:23:19....	0
"2021-08-18 16:27:18....	"2021-08-18 16:27:23....	"WORK05"	"LINE001"	"LOT132"	1577	"2021-08-18T08:23:19....	0
"2021-08-18 16:27:08....	"2021-08-18 16:27:18....	"WORK04"	"LINE001"	"LOT132"	1576	"2021-08-18T08:23:18....	0
"2021-08-18 16:27:06....	"2021-08-18 16:27:07....	"WORK03"	"LINE001"	"LOT132"	1575	"2021-08-18T08:23:18....	0
"2021-08-18 16:27:05....	"2021-08-18 16:27:06....	"WORK02"	"LINE001"	"LOT132"	1574	"2021-08-18T08:23:18....	0

Stream Analytics job の開始

Stream Analyticsの画面にて、以下の順でボタンを押下します。

① [概要]を選択

② [開始]をクリック



※通常、1分～3分程度で
開始されます。

5. サーバーレス SQL プールによるデータ参照

サーバーレス SQL プールの特徴

特徴

- データ ウェアハウス機能
- スケールアウト型の超並列データ分析エンジン
- T-SQL のサポート (SQL Server との高い互換性)
- Spark pool とのメタデータ共有

利点

- サーバーレス クエリ (データ処理量課金)
- 管理オーバーヘッドが無い
- データは Data Lake に存在し、データロード不要

適用領域

- Data Lake 上のデータに対する大規模な加工処理
- データ サイエンティスト向けの分析・加工プラットフォーム

Synapse Studio の起動

Synapse Analyticsの画面にて、以下の順でボタンを押下します。

- ① [概要]を選択
- ② [Synapse Studioを開く] → [オープン] を選択

The screenshot shows the Azure Synapse Analytics workspace overview page for 'syn-handa-handson'. A red box labeled ① highlights the '概要' (Overview) button in the left sidebar. Another red box labeled ② highlights the 'Synapse Studioを開く' (Open Synapse Studio) button in the '作業の開始' (Get Started) section at the bottom left.

syn-handa-handson

検索 (Ctrl+ /)

概要

リソース グループ (変更) : 20210611-handson

状態 : Succeeded

場所 : 東日本

サブスクリプション (変更) : Microsoft Azure Internal

サブスクリプション ID :

マネージド仮想ネットワ... : いいえ

マネージド ID オブジェ... :

ワークスペースの Web ... : <https://web.azure-synapse.net/ja/?workspace=%2fsubscriptions%2f976...>

タグ (変更) : タグを追加するにはここをクリック

Synapse Studio を開く

完全に統合された分析ソリューションの構築を開始し、新しい分析情報を利用可能にします。

ドキュメントを読む

迅速に生産性を高める方法について説明します。概念、チュートリアル、サンプルをご確認ください。

詳細情報

②

Data Lake Storage データの確認

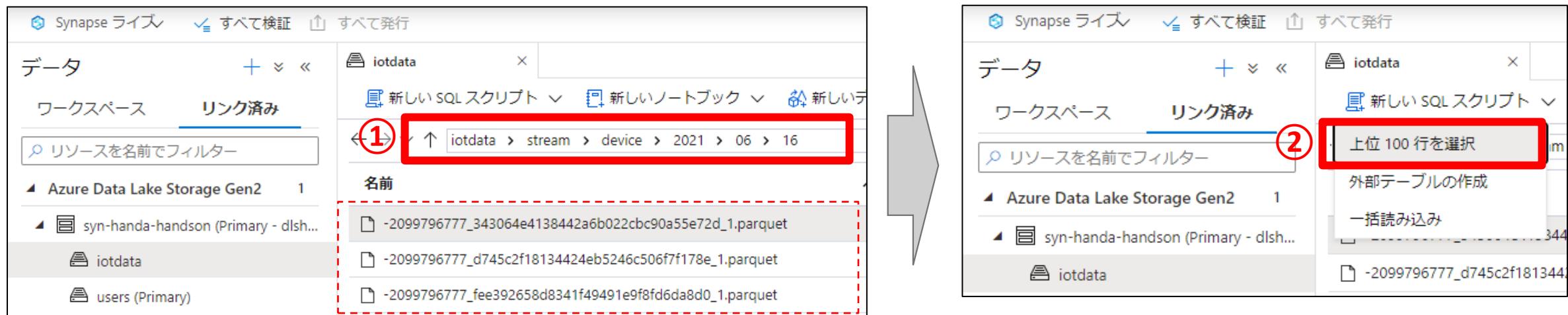
Synapse Analytics ワークスペースの画面にて、以下の順でボタンを押下します。

- ① [データ] を選択
- ② [リンク済み] を選択
- ③ [Azure Data Lake Storage Gen2] > [syn-<name>-handson] > [iotdata] を選択

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar has icons for Home, Development, Integration, Monitoring, and Management. The 'Data' icon is highlighted with a red box and circled with a red number 1. The main content area shows a 'Data' section with a 'Link Status' dropdown menu. The 'Linked' option is selected and highlighted with a red box and circled with a red number 2. Below this, there is a search bar and a list of Azure Data Lake Storage Gen2 resources. One resource, 'syn-handa-handson (Primary - dlsh...)', is expanded, showing its contents: 'iotdata' and 'users (Primary)'. The 'iotdata' folder is highlighted with a red box and circled with a red number 3. The top navigation bar includes the Microsoft Azure logo, the workspace name 'Synapse Analytics > syn-handa-handson', a search bar, and various status indicators.

サーバーレス SQL プールによる Parquet のクエリ

- ① [iotdata] の [stream] > [device] > [YYYY] > [MM] > [DD] へ移動
- ② 格納されているparquetファイルを選択し、[新しいSQLスクリプト] > [上位 100 行を選択] を選択



サーバーレス SQL プールによる Parquet のクエリ

[実行] をクリックし、クエリ結果が表示されることを確認

The screenshot shows the Azure Data Studio interface with the following details:

- Top Bar:** Shows the database name "iotdata", the script tab "SQL script 1", connection status, and the master database.
- Toolbar:** Includes buttons for "実行" (Run) with a red box highlighting it, "戻す" (Undo), "発行" (Publish), "クエリプラン" (Query Plan), "次に接続" (Next Connection), "組み込み" (Embedded), and "データベースの使用" (Database Usage).
- Query Editor:** Displays the following T-SQL code:

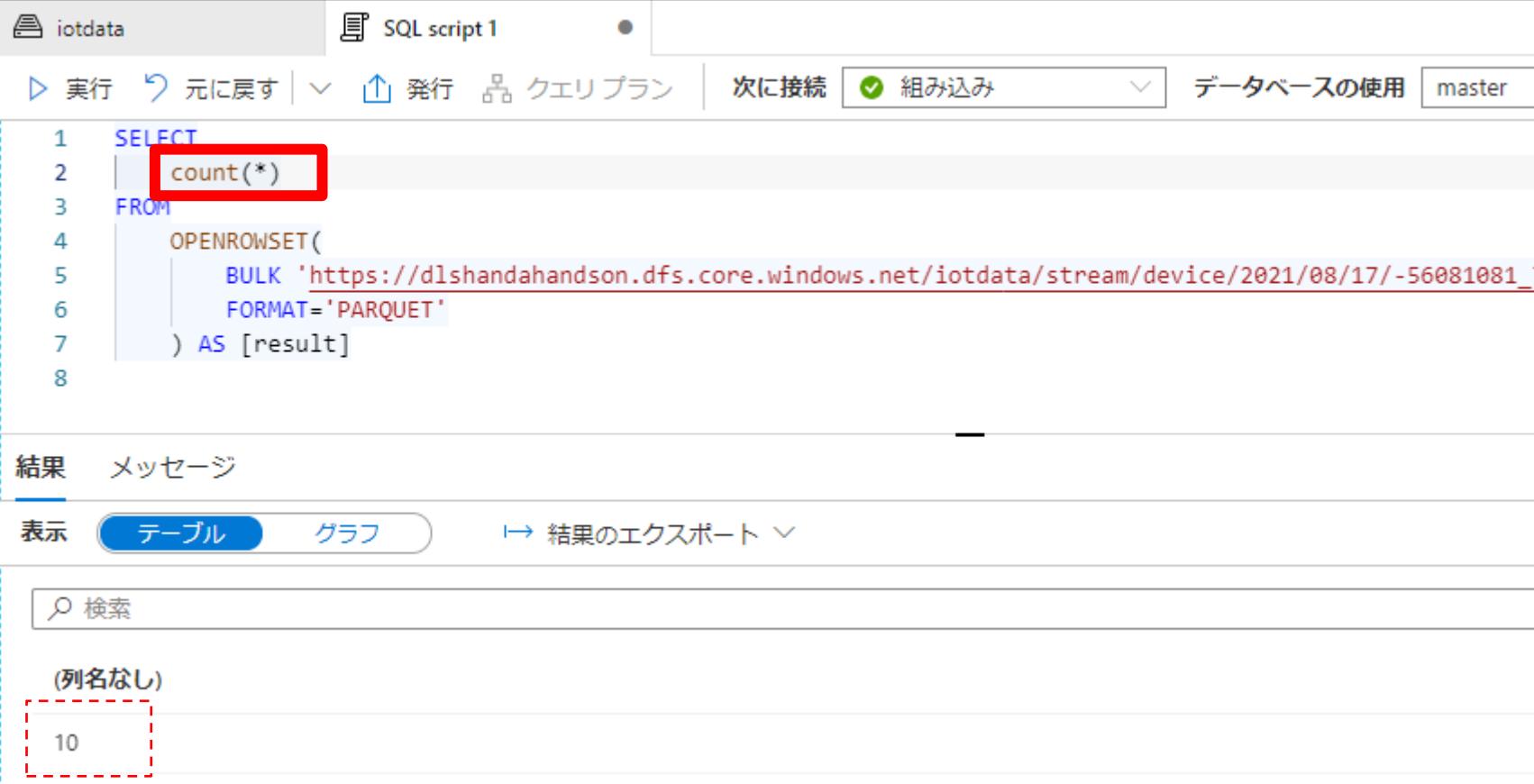
```
1 SELECT
2     TOP 100 *
3     FROM
4     OPENROWSET(
5         BULK 'https://dlshandahanson.dfs.core.windows.net/iotdata/stream/device/2021/08/17/-56081081\_7a93a448c9e94eabad9c989cceff',
6             FORMAT='PARQUET'
7     ) AS [result]
```
- Results Tab:** Shows the "結果" (Results) tab selected. The results are displayed as a table with the following columns: Start_time, End_time, Work, Line_num, Lot_num, Serial_num, EventProcesse..., PartitionId, and EventEnqu... . The data rows show 10 entries from 2021-08-18 00:00:00 to 2021-08-18 00:00:00 for WORK01 through WORK10.
- Table View:** The results are shown in a table format with 10 rows and 9 columns.

Start_time	End_time	Work	Line_num	Lot_num	Serial_num	EventProcesse...	PartitionId	EventEnqu...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK01	LINE001	LOT033	386	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK02	LINE001	LOT033	387	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK03	LINE001	LOT033	388	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK04	LINE001	LOT033	389	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK05	LINE001	LOT033	390	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK06	LINE001	LOT033	391	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK07	LINE001	LOT033	392	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK08	LINE001	LOT033	393	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK09	LINE001	LOT033	394	2021-08-17T15...	0	2021-08-1...
2021-08-18 00:00:00	2021-08-18 00:00:00	WORK10	LINE001	LOT033	395	2021-08-17T15...	0	2021-08-1...

サーバーレス SQL プールによる Parquet のクエリ

該当のparquetファイルに含まれる件数を取得

(「TOP 100 *」の箇所を「count(*)」に変更して実行)



The screenshot shows the Azure Data Studio interface with the following details:

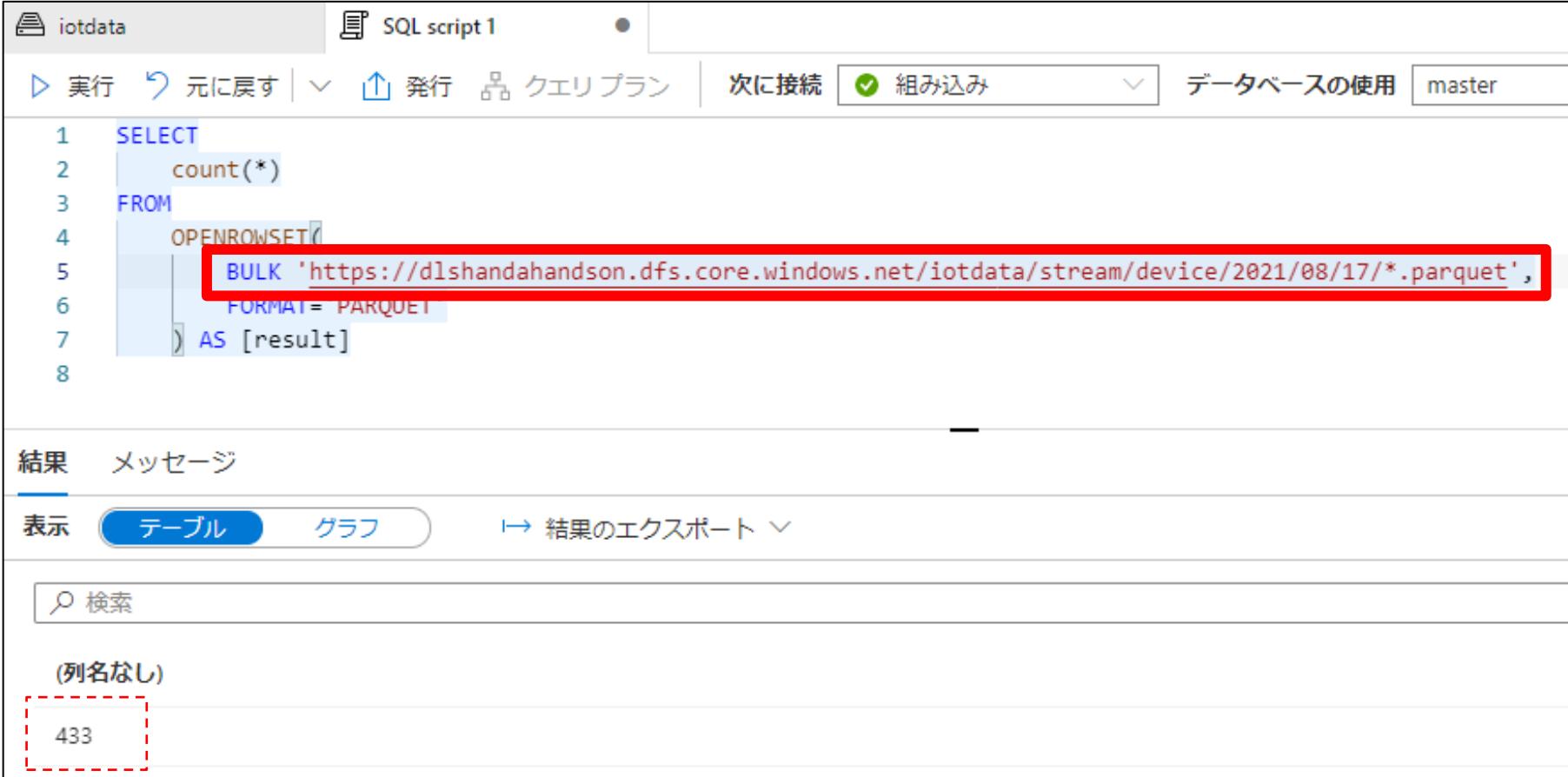
- Top Bar:** Shows the database name "iotdata" and the script name "SQL script 1".
- Toolbar:** Includes buttons for "実行" (Run), "元に戻す" (Undo), "発行" (Publish), "クエリプラン" (Query Plan), "次に接続" (Next Connection), "組み込み" (Embedded), and "データベースの使用" (Database Usage) set to "master".
- Query Editor:** Displays the following T-SQL code:

```
1 SELECT
2     count(*)
3 FROM
4     OPENROWSET(
5         BULK 'https://dlshandahanson.dfs.core.windows.net/iotdata/stream/device/2021/08/17/-56081081\_7',
6         FORMAT='PARQUET'
7     ) AS [result]
```
- Results Tab:** Shows the result of the query: "(列名なし)" followed by the number "10".
- Bottom Navigation:** Includes tabs for "結果" (Results) and "メッセージ" (Messages), and buttons for "表示" (View), "テーブル" (Table), and "グラフ" (Graph). There is also a "結果のエクスポート" (Export Results) button.

サーバーレス SQL プールによる Parquet のクエリ

フォルダ内のすべてのparquetファイルに含まれる件数を取得

(ファイル名を直接指定する箇所を「*.parquet」に変更して実行)



The screenshot shows the Azure Data Studio interface with the following details:

- WindowTitle:** iodata - SQL script 1
- Toolbar Buttons:** 実行 (Run), 元に戻す (Undo), 発行 (Publish), クエリプラン (Query Plan), 次に接続 (Next Connection), 組み込み (Embedded) checked, データベースの使用 (Database Used) master.
- SQL Script Content:**

```
1 SELECT
2     count(*)
3 FROM
4     OPENROWSET(
5         BULK 'https://dlshandahanson.dfs.core.windows.net/iotdata/stream/device/2021/08/17/*.parquet',
6         FORMAT = PARQUET
7     ) AS [result]
8
```
- Result Tab:** 結果 (Results) selected, メッセージ (Messages) tab also visible.
- Display Options:** 表示 (View) dropdown with テーブル (Table) selected, グラフ (Graph) button, and 結果のエクスポート (Export Results) button.
- Search Bar:** 検索 (Search) input field.
- Output Area:** (列名なし) (No column names) displayed, followed by the value 433.

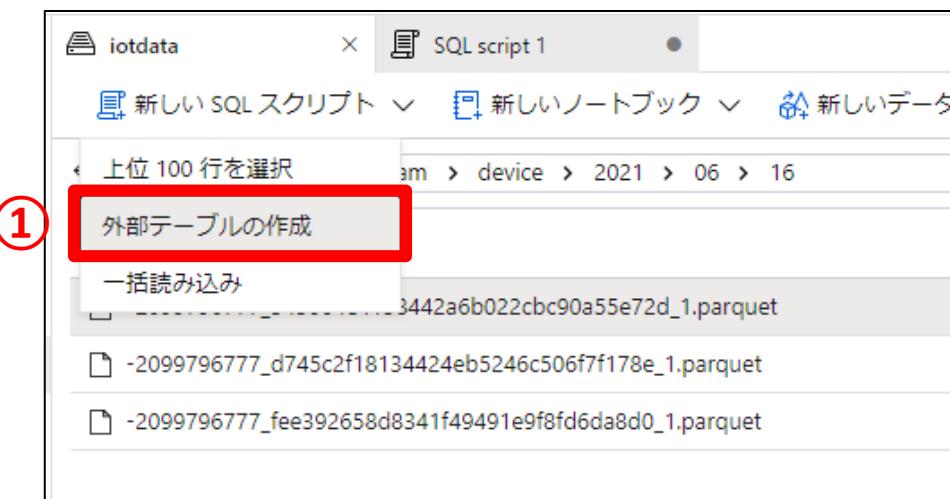
サーバーレス SQL プールで外部テーブルを作成

① 任意のparquetファイルを選択し、[新しいSQLスクリプト] > [外部テーブルの作成] を選択

② 以下の項目を設定

- ・SQL プールの選択：組み込み
- ・データベースの選択：任意（例：**serverlessdb**）
- ・外部テーブル名：任意（例：**demo**）

③ [作成] をクリック



The screenshot shows the 'Create External Table' configuration dialog. A red dashed box highlights the 'SQL Pool Selection' dropdown, which is set to 'SQL Pool Selection'. Another red dashed box highlights the 'Database Selection' dropdown, which is set to 'serverlessdb'. A third red dashed box highlights the 'External Table Name' input field, which is set to 'demo'. A red circle labeled '②' is placed over the 'SQL Pool Selection' dropdown. A red circle labeled '③' is placed over the 'Create' button at the bottom left of the dialog. A large gray arrow points from the left interface to this configuration dialog.

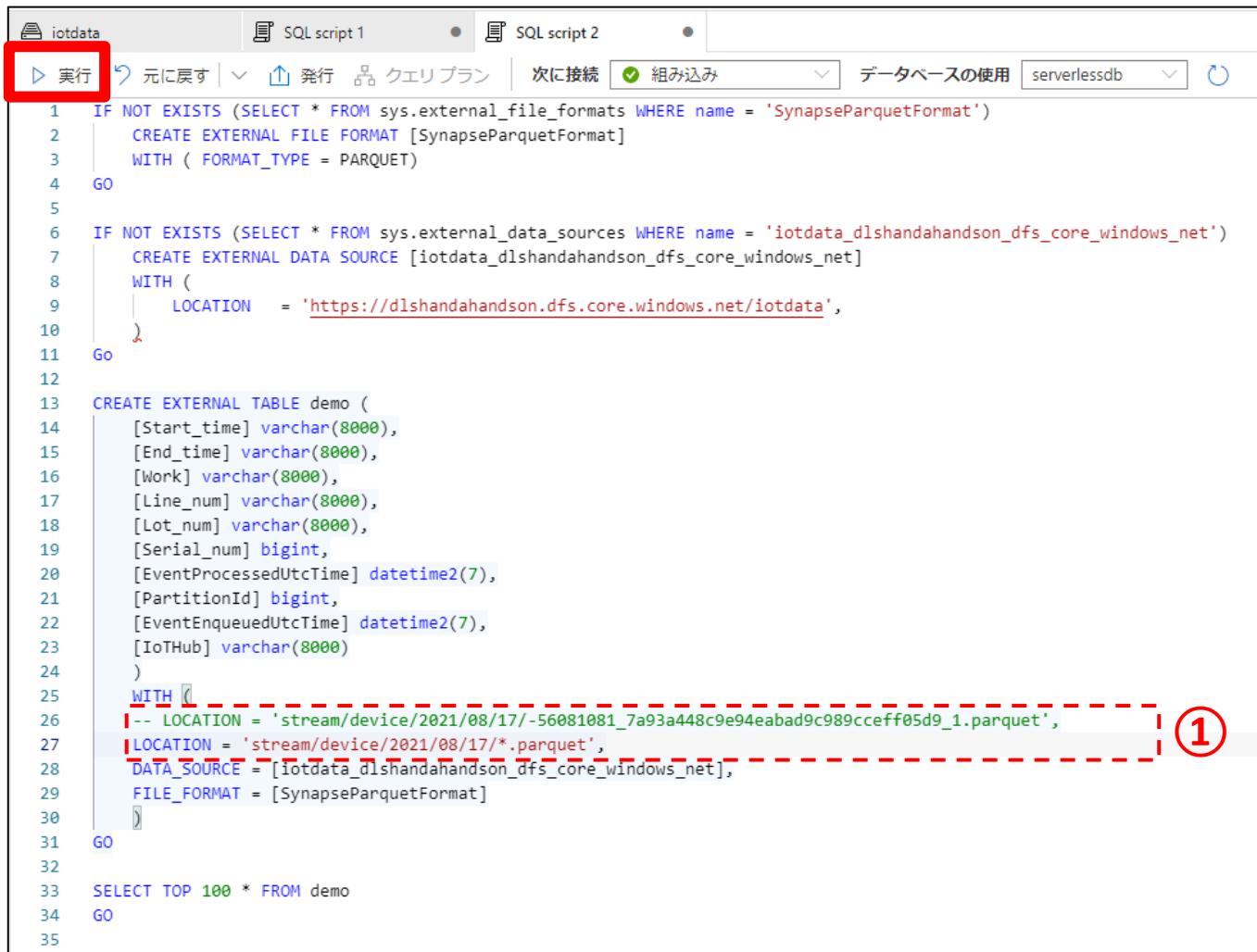
サーバーレス SQL プールで外部テーブルを作成

表示されるSQLスクリプトに対して、
以下の修正を行った上で、[実行] をクリック

Parquet はすべてのファイルを対象にするよう変更

<変更する行>

①
-- LOCATION = 'stream/device/2021/08/17/-
56081081_7a93a448c9e94eabad9c989cceff05d9_1.parquet',
LOCATION = 'stream/device/2021/08/17/*.parquet',



The screenshot shows the Azure Data Studio interface with a SQL script editor. The script is for creating an external file format and an external data source, and then defining an external table named 'demo'. A red box highlights the 'Execute' button at the top left of the editor. A red dashed box highlights the line of code being modified (line 26). A red circle with the number '1' is placed over the modified line.

```
1 IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormat')
2     CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
3     WITH ( FORMAT_TYPE = PARQUET)
4 GO
5
6 IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'iotdata_dlshandahanson_dfs_core_windows_net')
7     CREATE EXTERNAL DATA SOURCE [iotdata_dlshandahanson_dfs_core_windows_net]
8     WITH (
9         LOCATION = 'https://dlshandahanson.dfs.core.windows.net/iotdata',
10    )
11 Go
12
13 CREATE EXTERNAL TABLE demo (
14     [Start_time] varchar(8000),
15     [End_time] varchar(8000),
16     [Work] varchar(8000),
17     [Line_num] varchar(8000),
18     [Lot_num] varchar(8000),
19     [Serial_num] bigint,
20     [EventProcessedUtcTime] datetime2(7),
21     [PartitionId] bigint,
22     [EventEnqueuedUtcTime] datetime2(7),
23     [IoTHub] varchar(8000)
24 )
25 WITH (
26     -- LOCATION = 'stream/device/2021/08/17/-56081081_7a93a448c9e94eabad9c989cceff05d9_1.parquet',
27     LOCATION = 'stream/device/2021/08/17/*.parquet',
28     DATA_SOURCE = [iotdata_dlshandahanson_dfs_core_windows_net],
29     FILE_FORMAT = [SynapseParquetFormat]
30 )
31 Go
32
33 SELECT TOP 100 * FROM demo
34 GO
35
```

6. Power BI レポートの作成

重要なビジネスデータ分析も各従業員が可能に データ活用の民主化

社内に眠っているデータを活用し、多面的な分析による新たな“改革”をもたらすことが可能

これまでのBI

データ分析は専門家（一部の人）の仕事

セルフサービスBI

個々の従業員が簡単にデータ分析を行えるようになり、より高いパフォーマンスを発揮



このソリューションにより実現できること

業績データをより詳細に分析
取り組みを強める/開発する領域を発掘



データを活用した新しい提案の模索



オフィスワーカーだけではなく
現場の働き方にも踏み込んだ改革



可視化したデータによる
精度を上げた情報共有と意識の統一

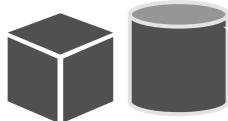
Power BI とは

誰もが容易にデータ活用を行い、
必要に応じて共有することを目的としたツールとプラットフォーム

蓄積されたデータ群



自社内のデータ



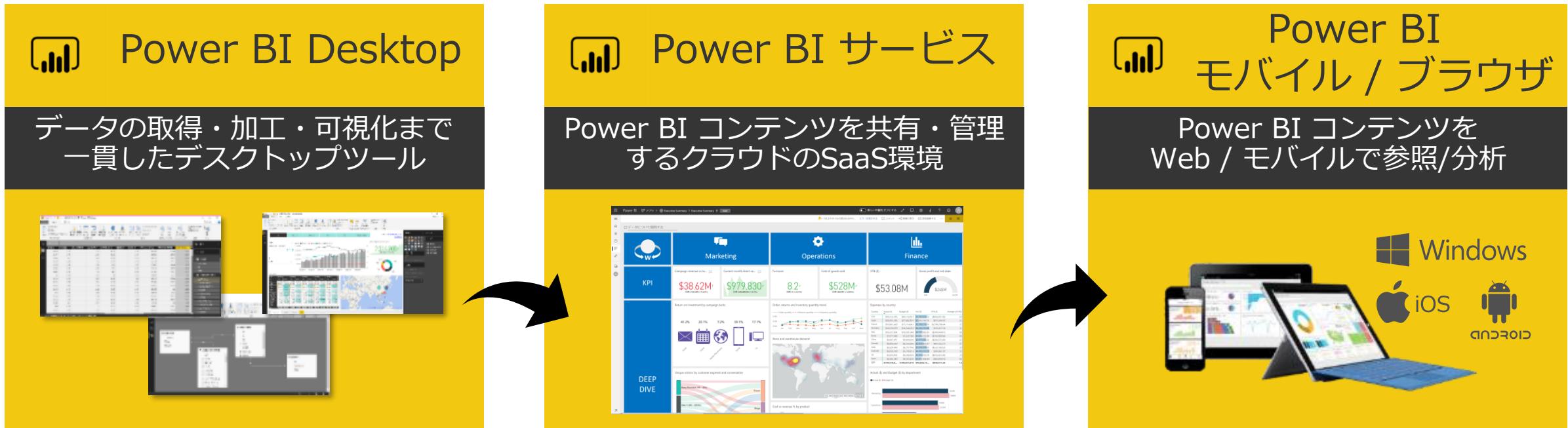
クラウド上のデータ



多様なデバイスでアクセス



Power BI の全体像



→
共有・展開

作成



基盤



←
参照・分析

閲覧 / 分析

レポート作成

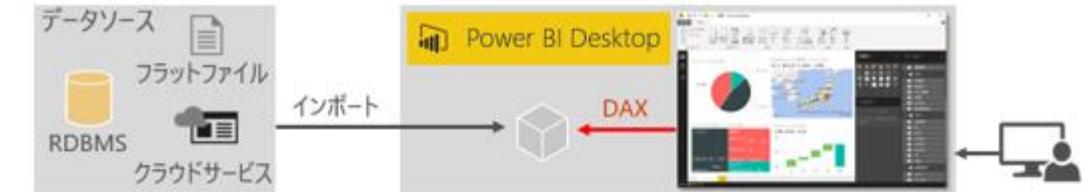
レポート共有基盤

接続方式の使い分け

インポート

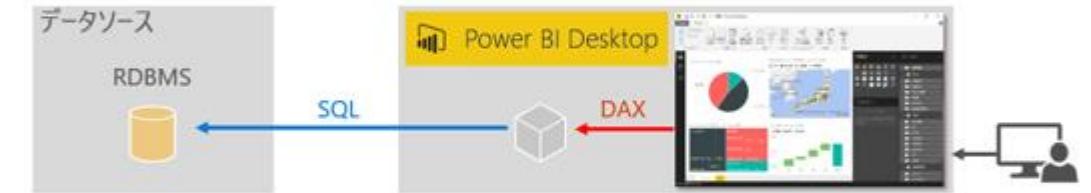
- ・ Power BIにデータ保持
- ・ 圧縮後1GBまでのデータ容量が選択目安
- ・ Power BI サービスにてスケジュール更新定義
- ・ Power Q&A、クイック分析を使用したカスタム分析可能

データ量 小



Direct Query

- ・ Power BIからデータを取得（データを保持しない）
- ・ 常に最新のデータ取得が可能



ライブ接続

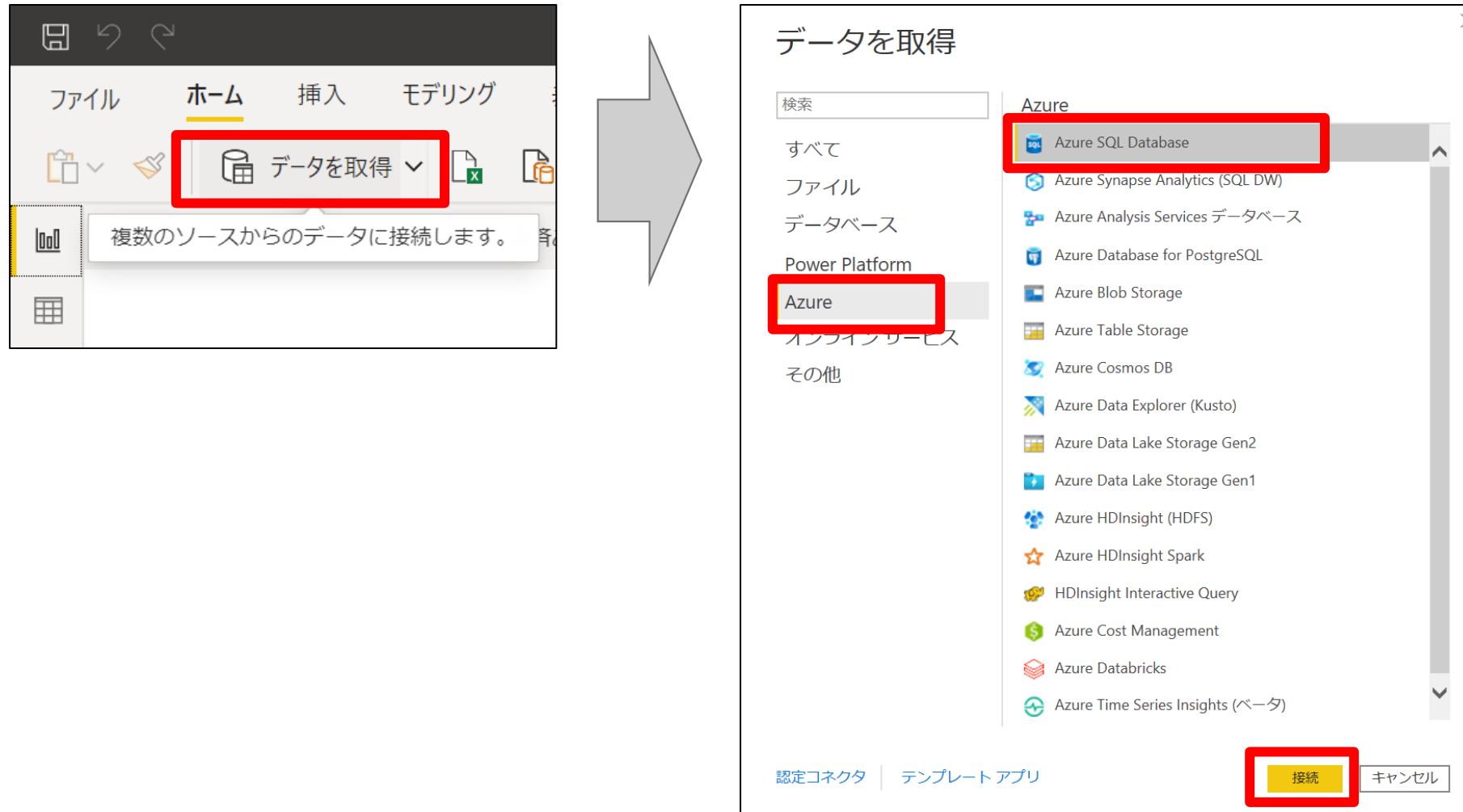
- ・ Power BIからデータを取得（データを保持しない）
- ・ 常に最新のデータ取得が可能
- ・ ログインユーザー情報でデータソースにアクセス

データ量 大



Power BI を起動し、データを取得

- Power BI Desktop を起動し、[ホーム] > [データを取得] をクリック
- [Azure] > [Azure SQL Database] を選択し、[接続] をクリック



サーバーレス SQL プールの指定

- [サーバー] 欄には Synapse Analytics の [概要] > [サーバーレスSQLエンドポイント] の値をコピー
- 他はデフォルトのまま [OK] をクリック

SQL Server データベース

サーバー ①
a-handson-ondemand.sql.azuresynapse.net

データ接続モード ①
 インポート
 DirectQuery

▷ 詳細設定オプション

OK キャンセル

syn-handson

Synapse ワークスペース

+ 新しい専用 SQL プール + 新しい Apache Spark プール 更新 SQL 管理者パスワードのリセット 削除

概要

リソース グループ (変更) : 20210611-handson
状態 : Succeeded
場所 : 東日本
サブスクリプション (変更) : Microsoft Azure Internal
サブスクリプション ID :
マネージド仮想ネットワ... : いいえ
マネージド ID オブジェ... :
ワークスペースの Web ... : https://web.azuresynapse.net/ja/?workspace=%2bsubscriptions%2f9...
タグ (変更) : タグを追加するにはここをクリック

設定

SQL Active Directory 管理者
プロパティ
ロック

Analytics プール

SQL プール
Apache Spark プール

セキュリティ

作業の開始

Synapse Studio を開く
完全に統合された分析ソリューションの構築を開始し、新しい分析情報を利用可能にします。
[オープン]

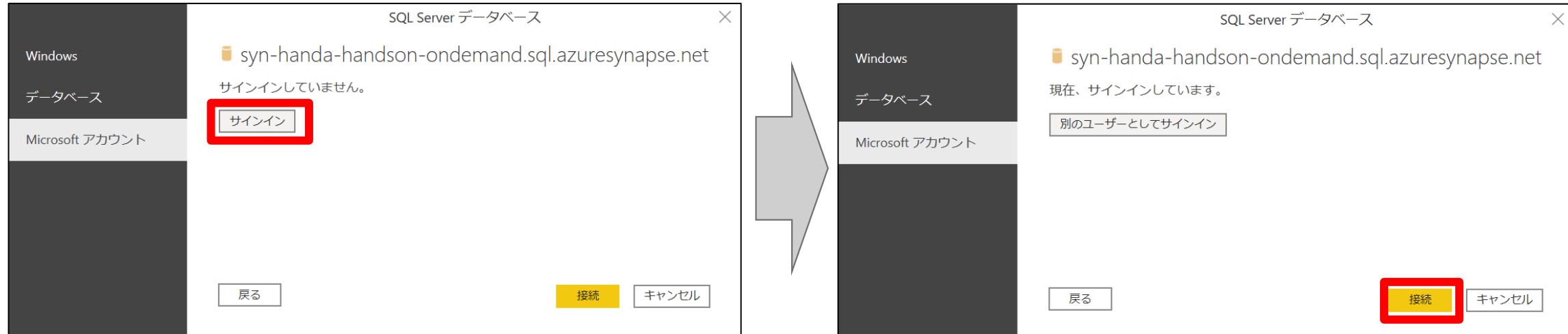
ドキュメントを読む
迅速に生産性を高める方法について説明します。概念、チュートリアル、サンプルをご確認ください。
[詳細情報]

ファイアウォール : ファイアウォール設定の表示
プライマリ ADLS Gen2 ... : https://dlshandahandson.dfs.core.windows.net
プライマリ ADLS Gen2 ... : users
SQL 管理ユーザー名 : sqladminuser
SQL Active Directory 管... : yahanda@microsoft.com
専用 SQL エンドポイント : syn-handson-handson.sql.azuresynapse.net
開発エンドポイント : https://syn-handson.dev.azuresynapse.net

サーバーレス SQL エン... : syn-handson-ondemand.sql.azuresynapse.net

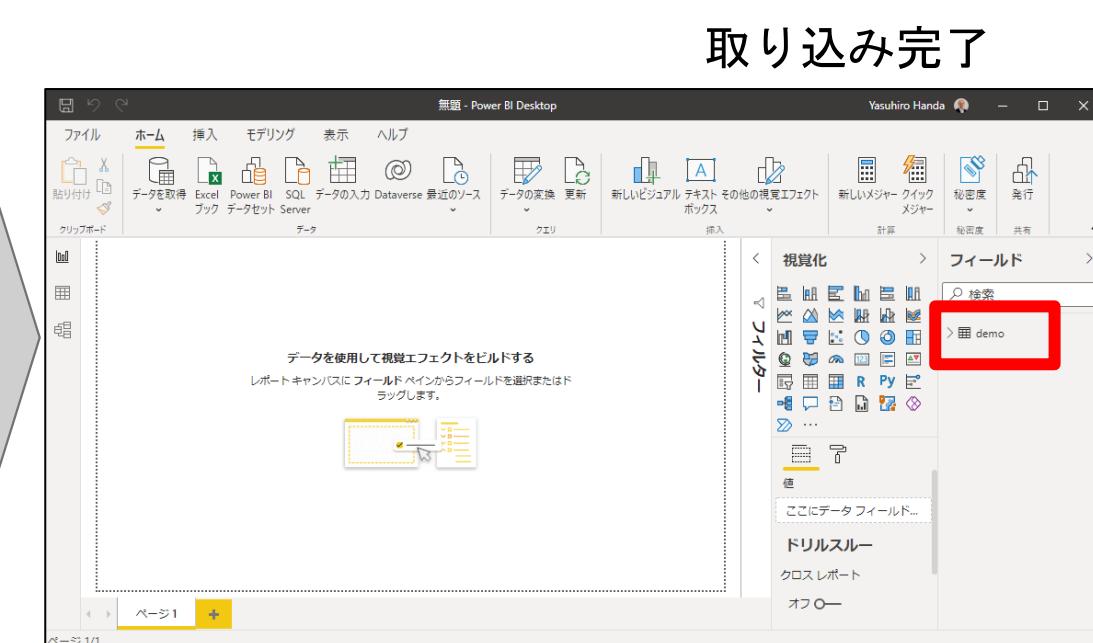
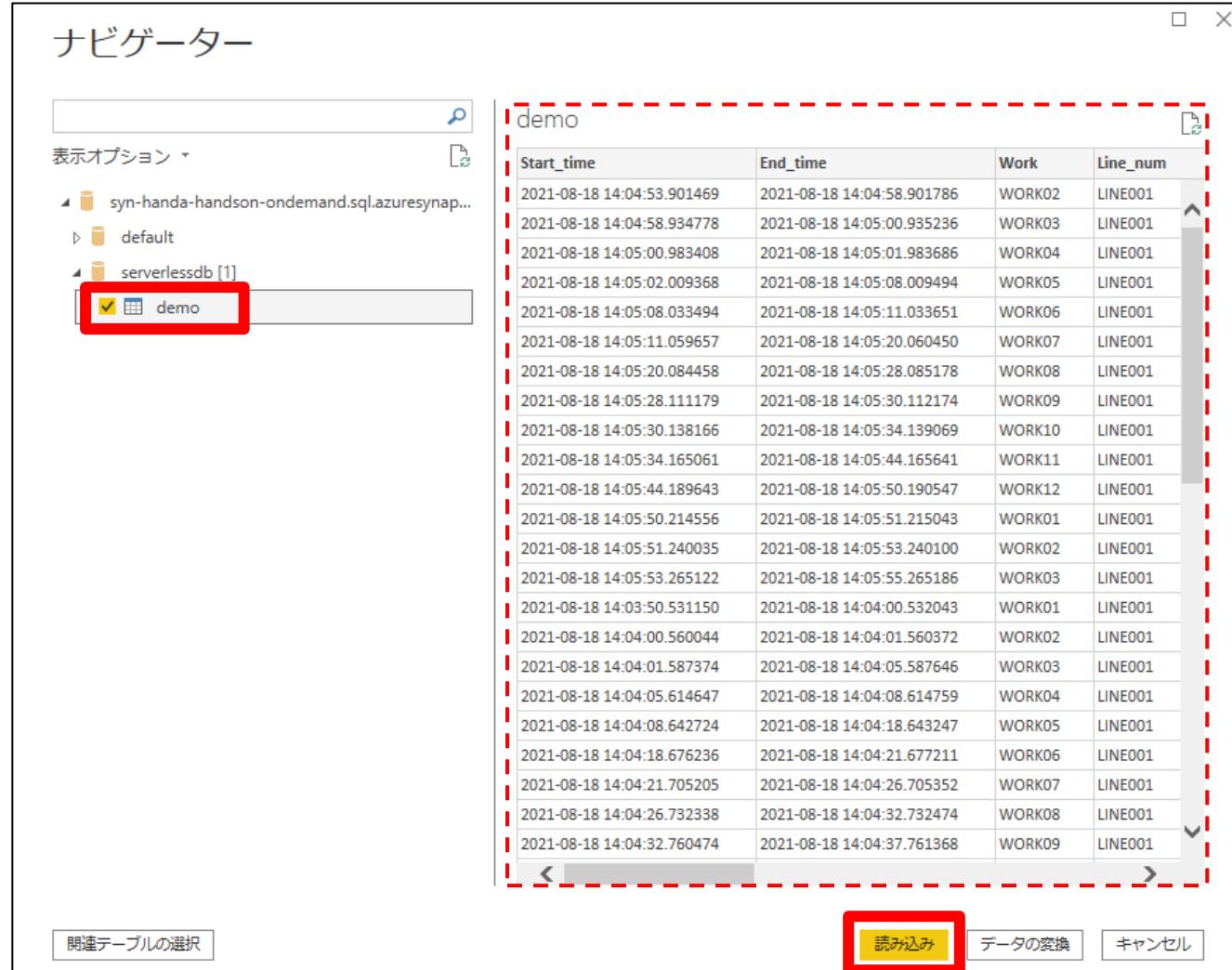
サーバーレス SQL プールへのサインイン

- [Microsoft アカウント] > [サインイン] をクリックし、ポップアップ画面よりサインインを実施
 - Azure ポータルのユーザーと同じユーザーでサインインすること
- サインインが完了したら [接続] をクリック



外部テーブルからのデータ読み込み

- 先の手順で作成した外部テーブルを選択し、[読み込み] をクリック



取り込み完了

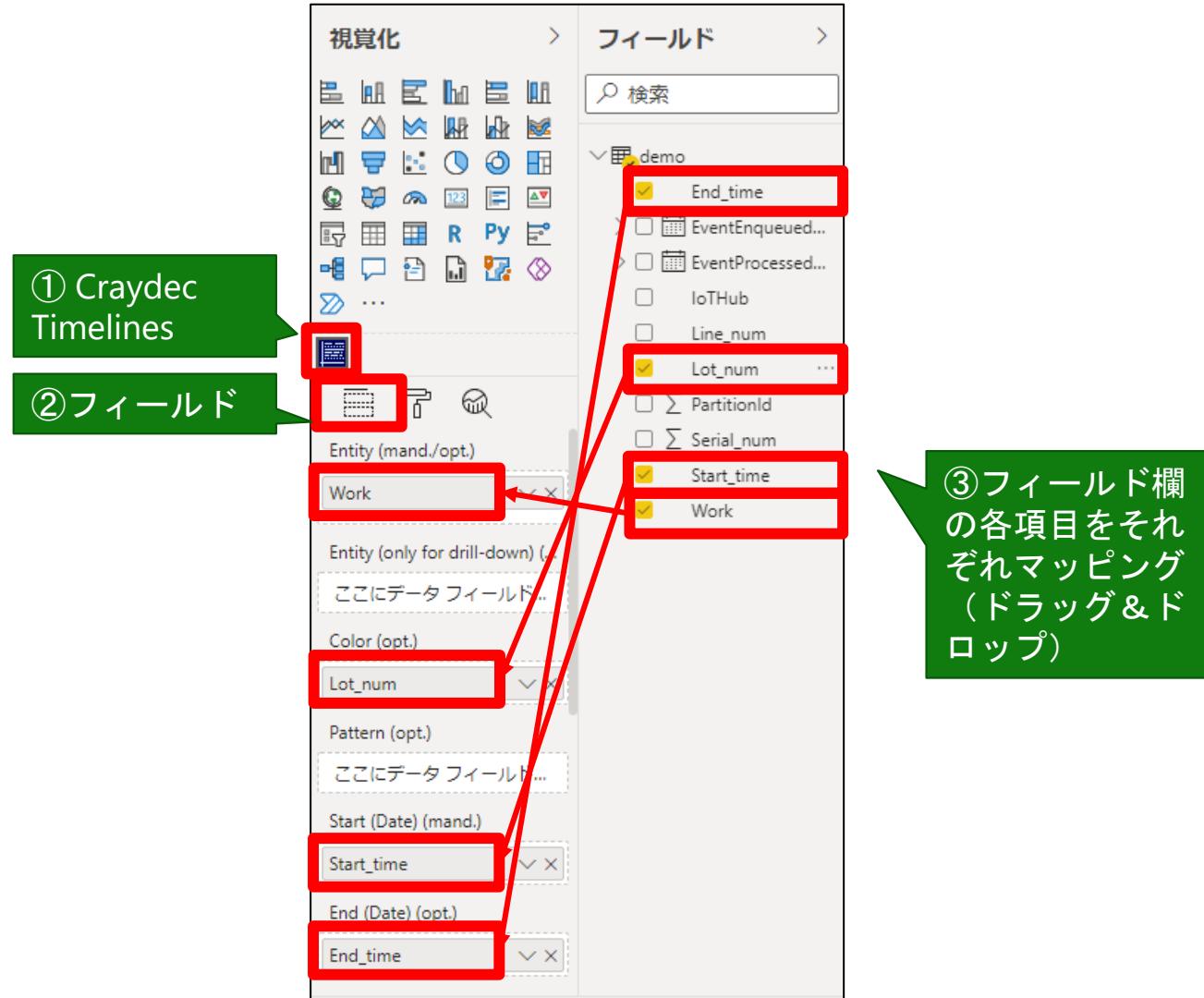
ビジュアルのダウンロード

- ・ 視覚化のメニューより、[...] > [その他のビジュアルの取得] をクリックします
- ・ [timeline] で検索を行い、[Craydec Timelines] をAppSourceよりダウンロードします



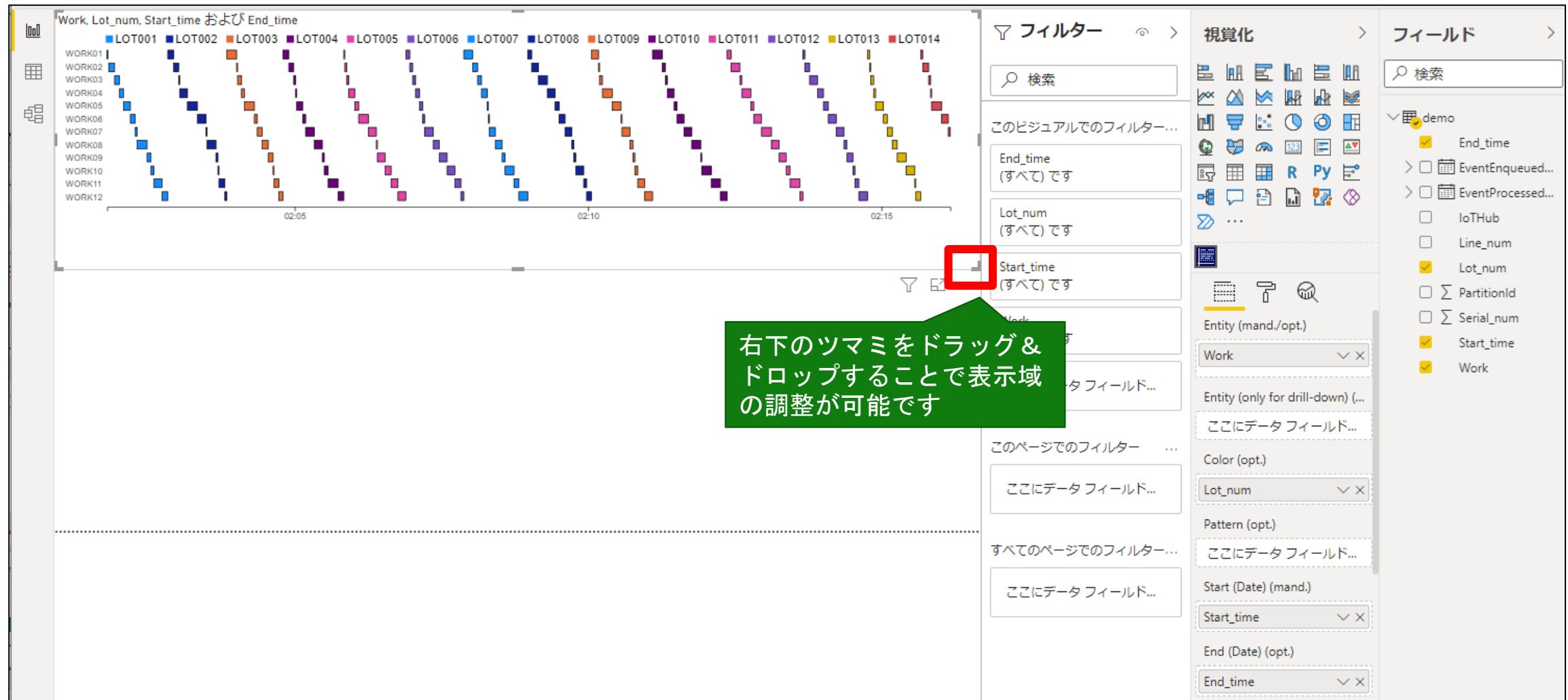
レポートの定義（各工程のサイクルタイム）

- ・ 視覚化のメニューより、インポートした [Craydec Timelines] を選択し、以下の通り設定を行います



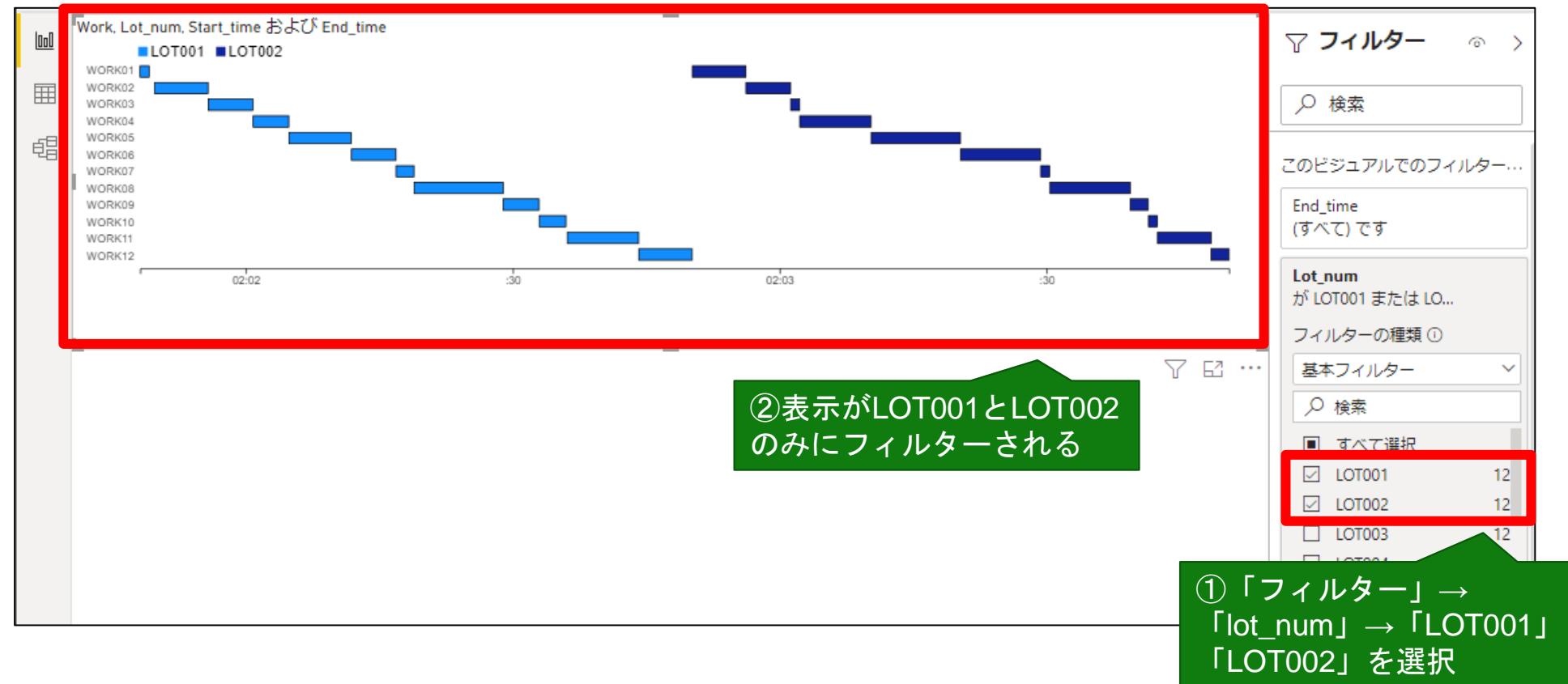
レポートの定義（各工程のサイクルタイム）

- ロット別のサイクルタイムが可視化されます



レポートの定義（各工程のサイクルタイム）

- 特定のロット番号でフィルターします



データの編集

- 各工程の作業時間を算出するため、取り込んだデータをPower BI Desktop上で編集します

The screenshot shows the Power BI Desktop interface. On the left, the 'Edit Queries' ribbon is visible, with the 'Data' icon highlighted by a green arrow and a red box. The main area shows a query editor for a table named 'demo'. The 'Table Tools' tab is selected. A green box highlights the 'New Column' button (grid icon) in the ribbon toolbar. The table view shows a list of rows with columns for Start_time, End_time, Work, Line_num, Lot_num, Serial_num, EventProcessedUtcTime, and Partition. The first few rows of data are:

Start_time	End_time	Work	Line_num	Lot_num	Serial_num	EventProcessedUtcTime	Partition
2021-08-18 14:10:57.800607	2021-08-18 14:11:05.801098	WORK12	LINE001	LOT009	108	8/18/2021 5:11:06 AM	
2021-08-18 14:11:05.826102	2021-08-18 14:11:15.826638	WORK01	LINE001	LOT010	109	8/18/2021 5:11:16 AM	
2021-08-18 14:11:15.850666	2021-08-18 14:11:18.850977	WORK02	LINE001	LOT010	110	8/18/2021 5:11:19 AM	
2021-08-18 14:11:18.888971	2021-08-18 14:11:20.889581	WORK03	LINE001	LOT010	111	8/18/2021 5:11:21 AM	
2021-08-18 14:11:20.917070	2021-08-18 14:11:27.917677	WORK04	LINE001	LOT010	112	8/18/2021 5:11:28 AM	
2021-08-18 14:11:27.948903	2021-08-18 14:11:30.949185	WORK05	LINE001	LOT010	113	8/18/2021 5:11:31 AM	
2021-08-18 14:11:30.975185	2021-08-18 14:11:37.975752	WORK06	LINE001	LOT010	114	8/18/2021 5:11:38 AM	
2021-08-18 14:11:38.001290	2021-08-18 14:11:46.001815	WORK07	LINE001	LOT010	115	8/18/2021 5:11:46 AM	
2021-08-18 14:11:46.025815	2021-08-18 14:11:55.026248	WORK08	LINE001	LOT010	116	8/18/2021 5:11:55 AM	
2021-08-18 14:11:55.053395	2021-08-18 14:11:56.053574	WORK09	LINE001	LOT010	117	8/18/2021 5:11:56 AM	

データの編集

- 開始時刻と終了時刻の差分を作業時間として算出し、新たに列を追加します

作業時間(秒) = DATEDIFF(

```
DATE(MID([Start_time], 1, 4), MID([Start_time], 6, 2), MID([Start_time], 9, 2)) + TIME(MID([Start_time], 12, 2), MID([Start_time], 15, 2), MID([Start_time], 18, 2)),  
DATE(MID([End_time], 1, 4), MID([End_time], 6, 2), MID([End_time], 9, 2)) + TIME(MID([End_time], 12, 2), MID([End_time], 15, 2), MID([End_time], 18, 2)),  
SECOND)
```

The screenshot shows the Microsoft Power BI Table Editor interface. At the top, there's a ribbon with 'ファイル' (File), 'ホーム' (Home), 'ヘルプ' (Help), 'テーブル ツール' (Table Tools), and '列ツール' (Column Tools). Below the ribbon, there's a toolbar with buttons for '名前' (Name), 'データ型' (Data Type), '構造体' (Structure), '書式設定' (Format), '要約処理' (Summary), 'データ カテゴリ' (Data Category), '並べ替え' (Sort), 'データ グループ' (Data Group), 'リレーションシップの管理' (Relationship Management), and '新しい列' (New Column). A green callout bubble points to the '名前' field with the text 'ここに上記式を入力してEnter'.

In the main area, there's a code editor window with a red border containing the following DAX formula:

```
1 作業時間(秒) = DATEDIFF(  
2   DATE(MID([Start_time], 1, 4), MID([Start_time], 6, 2), MID([Start_time], 9, 2)) + TIME(MID([Start_time], 12, 2), MID([Start_time], 15, 2), MID([Start_time], 18, 2)),  
3   DATE(MID([End_time], 1, 4), MID([End_time], 6, 2), MID([End_time], 9, 2)) + TIME(MID([End_time], 12, 2), MID([End_time], 15, 2), MID([End_time], 18, 2)),  
4   SECOND)
```

Below the code editor is a table view showing several rows of data. The columns are: Start_time, End_time, Work, Line_num, Lot_num, Serial_num, EventProcessedUtcTime, PartitionId, EventEnqueuedUtcTime, IoTHub, and the newly added '作業時間(秒)' column. The last column contains numerical values representing the calculated duration in seconds. A green callout bubble points to this column with the text '列が追加され、算出結果が表示される'.

Start_time	End_time	Work	Line_num	Lot_num	Serial_num	EventProcessedUtcTime	PartitionId	EventEnqueuedUtcTime	IoTHub	作業時間(秒)
2021-08-18 14:10:57.800607	2021-08-18 14:11:05.801098	WORK12	LINE001	LOT009	108	8/18/2021 5:11:06 AM	0	8/18/2021 5:11:06 AM	{"ConnectionDevice1"}	8
2021-08-18 14:11:05.826102	2021-08-18 14:11:15.826638	WORK01	LINE001	LOT010	109	8/18/2021 5:11:16 AM	0	8/18/2021 5:11:16 AM	{"ConnectionDevice1"}	10
2021-08-18 14:11:15.850666	2021-08-18 14:11:18.850977	WORK02	LINE001	LOT010	110	8/18/2021 5:11:19 AM	0	8/18/2021 5:11:19 AM	{"ConnectionDevice1"}	3
2021-08-18 14:11:18.888971	2021-08-18 14:11:20.889581	WORK03	LINE001	LOT010	111	8/18/2021 5:11:21 AM	0	8/18/2021 5:11:21 AM	{"ConnectionDevice1"}	2
2021-08-18 14:11:20.917070	2021-08-18 14:11:27.917677	WORK04	LINE001	LOT010	112	8/18/2021 5:11:28 AM	0	8/18/2021 5:11:28 AM	{"ConnectionDevice1"}	7
2021-08-18 14:11:27.948903	2021-08-18 14:11:30.949185	WORK05	LINE001	LOT010	113	8/18/2021 5:11:31 AM	0	8/18/2021 5:11:31 AM	{"ConnectionDevice1"}	3
2021-08-18 14:11:30.975185	2021-08-18 14:11:37.975752	WORK06	LINE001	LOT010	114	8/18/2021 5:11:38 AM	0	8/18/2021 5:11:38 AM	{"ConnectionDevice1"}	7
2021-08-18 14:11:38.001290	2021-08-18 14:11:46.001815	WORK07	LINE001	LOT010	115	8/18/2021 5:11:46 AM	0	8/18/2021 5:11:46 AM	{"ConnectionDevice1"}	8
2021-08-18 14:11:46.025815	2021-08-18 14:11:55.026248	WORK08	LINE001	LOT010	116	8/18/2021 5:11:55 AM	0	8/18/2021 5:11:55 AM	{"ConnectionDevice1"}	9
2021-08-18 14:11:55.053395	2021-08-18 14:11:56.053574	WORK09	LINE001	LOT010	117	8/18/2021 5:11:56 AM	0	8/18/2021 5:11:56 AM	{"ConnectionDevice1"}	1

データの編集

- 同様の手順で、次の二つの列も追加します

平均超過作業時間(秒) =

```
IF([作業時間(秒)] > AVERAGE([作業時間(秒)]),  
[作業時間(秒)] - AVERAGE([作業時間(秒)]),  
0  
)
```

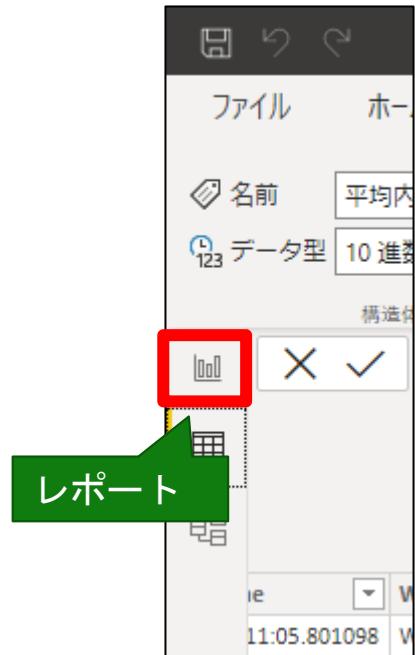
平均内作業時間(秒) =

```
IF([作業時間(秒)] <= AVERAGE([作業時間(秒)]),  
[作業時間(秒)],  
AVERAGE([作業時間(秒)])  
)
```

	作業時間(秒)	平均超過作業時間(秒)	平均内作業時間(秒)
viceld'	8	2.74233128834356	5.25766871165644
viceld'	10	4.74233128834356	5.25766871165644
viceld'	3	0	3
viceld'	2	0	2
viceld'	7	1.74233128834356	5.25766871165644
viceld'	3	0	3
viceld'	7	1.74233128834356	5.25766871165644
viceld'	8	2.74233128834356	5.25766871165644
viceld'	9	3.74233128834356	5.25766871165644
viceld'	1	0	1
viceld'	6	0.742331288343558	5.25766871165644
viceld'	5	0	5
viceld'	4	0	4
viceld'	7	1.74233128834356	5.25766871165644
viceld'	5	0	5
viceld'	2	0	2
viceld'	10	4.74233128834356	5.25766871165644
viceld'	4	0	4

データの編集

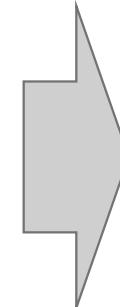
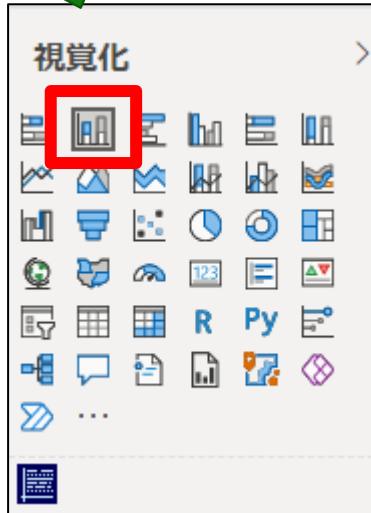
- 元のレポート画面に戻ります



レポートの定義（各作業時間の時間推移）

- 「積み上げ縦棒グラフ」を使って各作業時間の時間推移を表示します

①「積み上げ縦棒グラフ」を選択



The screenshot shows the 'Fields' mapping interface. On the left, there's a 'Visualizations' pane with various chart icons. On the right, the 'Fields' pane displays a tree structure under a 'demo' folder. The 'Axis' section contains a dropdown menu with 'Lot_num' selected. The 'Value' section contains two dropdown menus, both with '平均内作業時間(秒)' and '平均超過作業時間(秒)' selected. Red boxes highlight the 'Lot_num' dropdown in the Axis section and the two 'Value' dropdowns in the Value section. A green callout on the right points to the 'Fields' pane, and another red callout points to the 'Lot_num' dropdown.

②フィールド欄の各項目をそれぞれ上記の通りマッピング（ドラッグ＆ドロップ）

レポートの定義（各作業時間の時間推移）

- 書式の設定を行います

書式

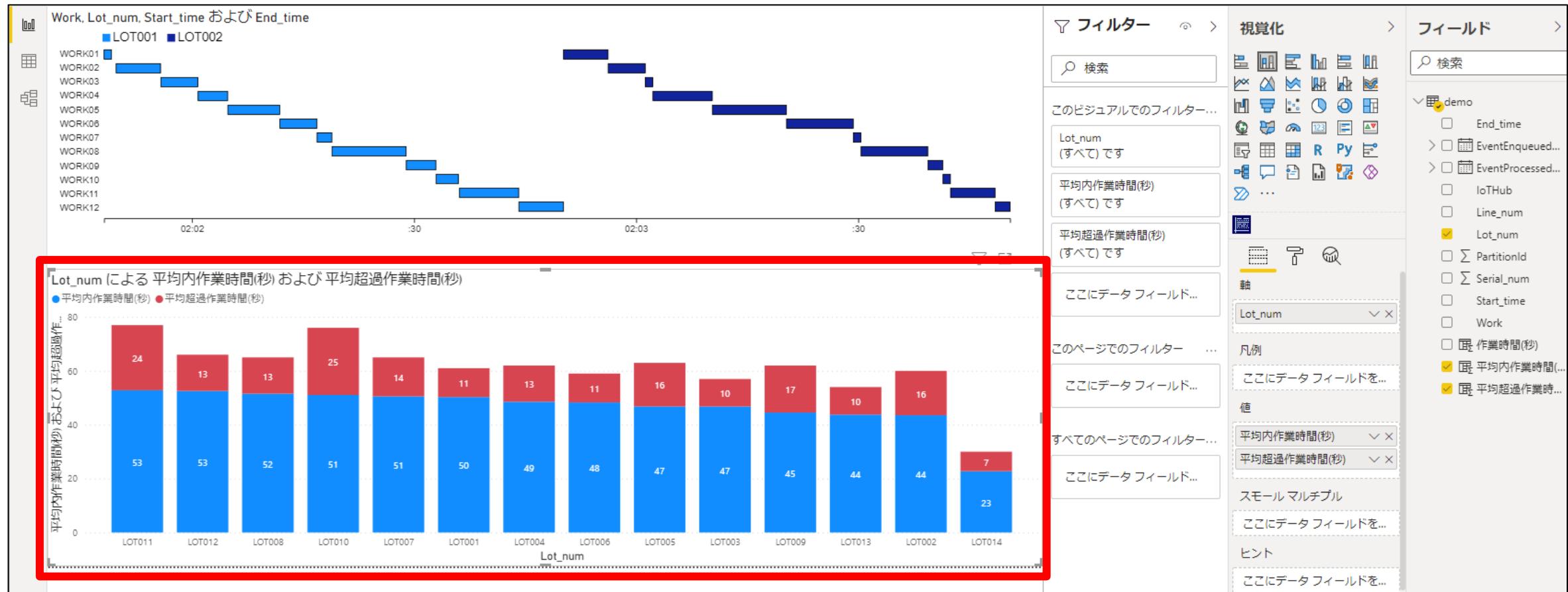
③ 「データの色」 → 「平均超過作業時間(秒)」を赤色に変更

④ 「データラベル」を「オン」に設定

⑤ 「オーバーフロー テキスト」を「オン」に設定

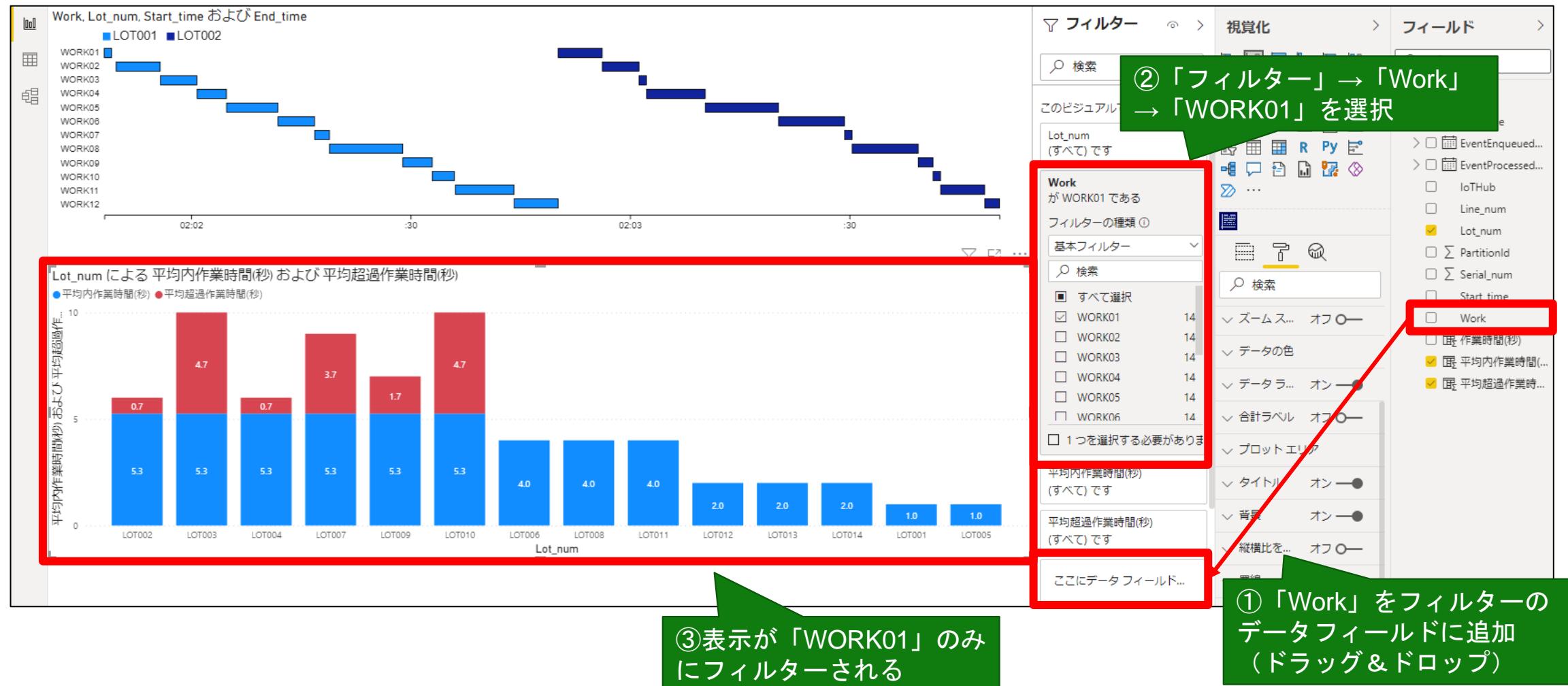
レポートの定義（各作業時間の時間推移）

- 「積み上げ縦棒グラフ」を先ほど作成したタイムラインの下に配置します



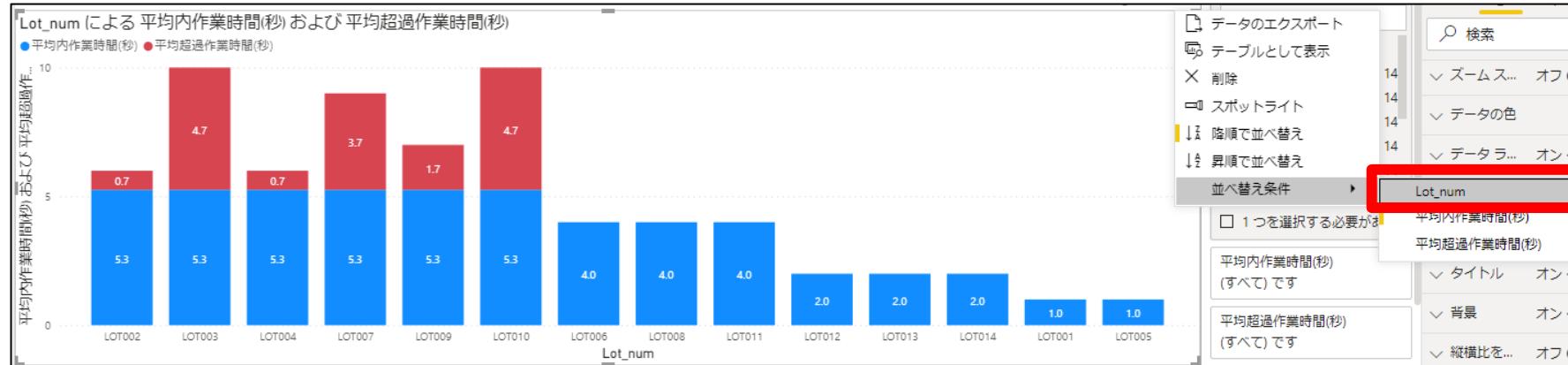
レポートの定義（各作業時間の時間推移）

- 特定の作業種別でフィルターします

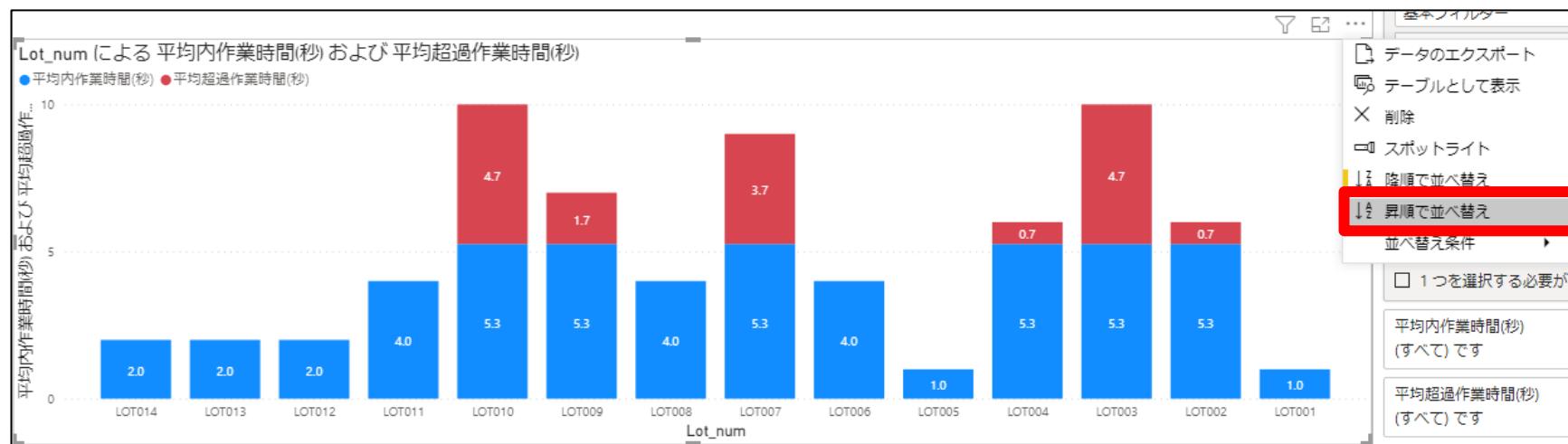


レポートの定義（各作業時間の時間推移）

- 作業時間の降順で表示されているため、ロット番号の昇順に並び替えます



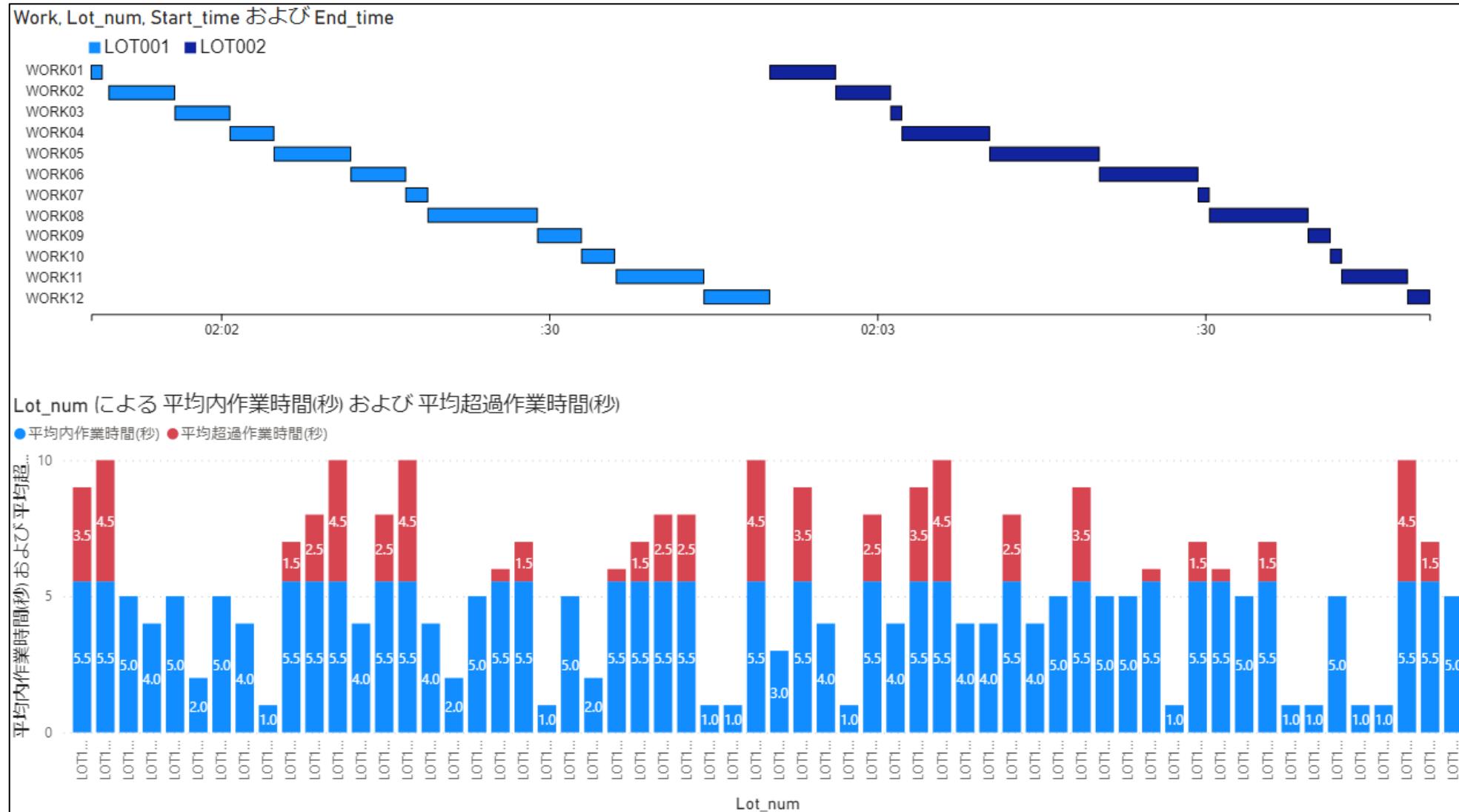
① 「...」 → 「並べ替え条件」 → 「Lot_num」を選択



② 「...」 → 「昇順で並べ替え」を選択

レポートの操作／相互作用の変更

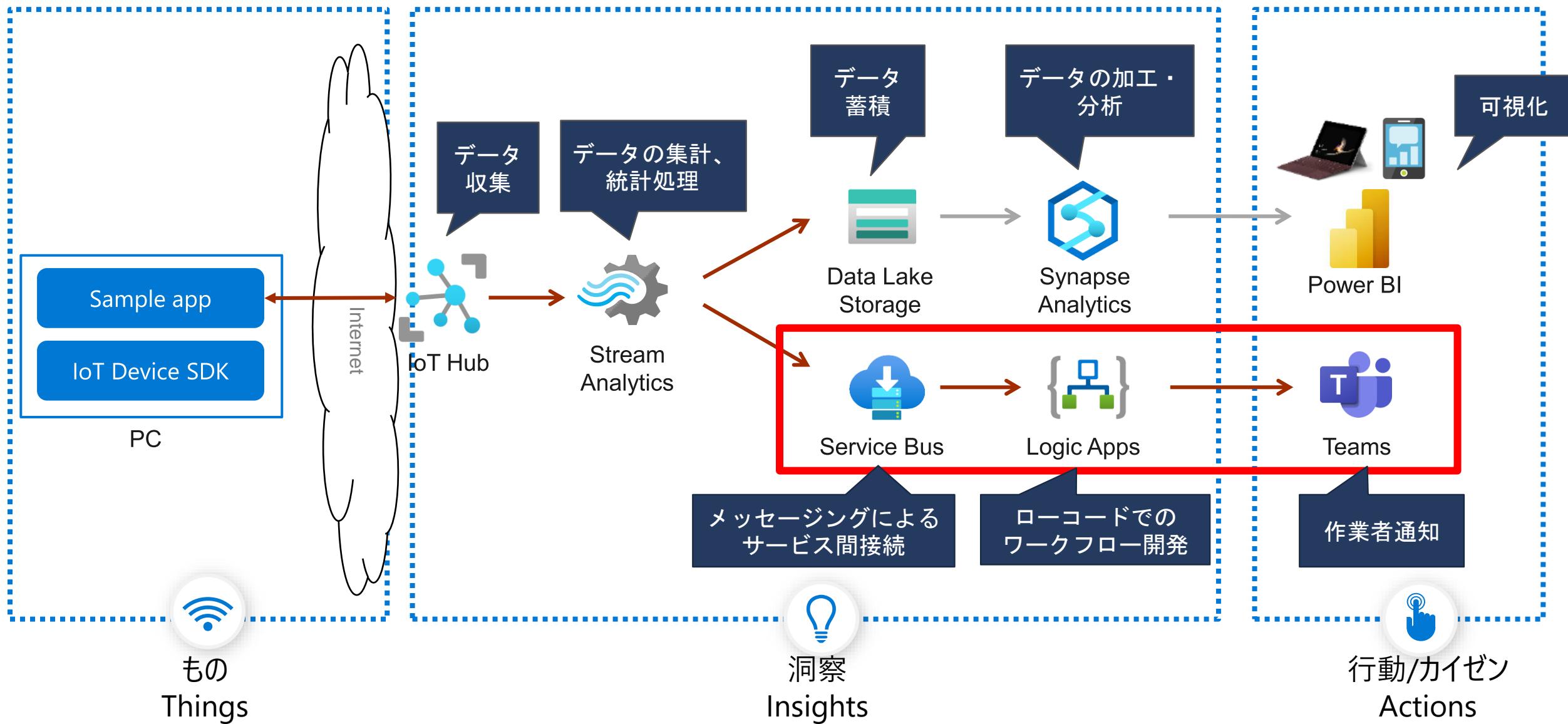
- ビジュアル内の各項目をクリックし、レポートを操作してみましょう。



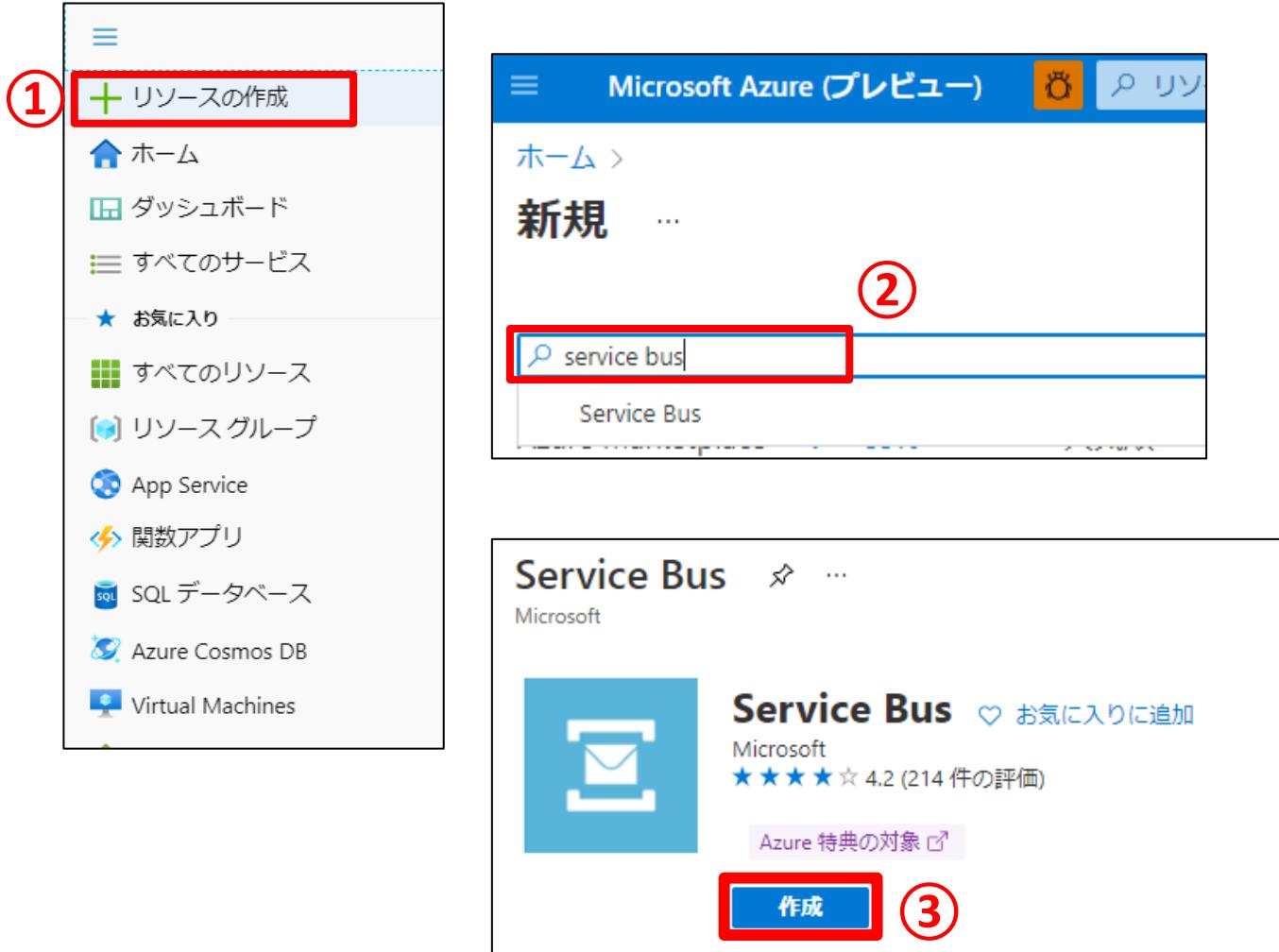
7. Teams アラート連携

ハンズオン構成

→ バッチ
→ リアルタイム



Service Bus の作成



1. Azure Portalへのアクセス
<https://ms.portal.azure.com>

2. 以下の順にメニューを押下
- ① [+ リソースの作成]
 - ② [Service Bus] を検索
 - ③ [作成] を選択

Service Bus の各種設定

名前空間の作成 ...
Service Bus

基本 タグ 確認および作成

プロジェクトの詳細
デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソースグループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション* Microsoft Azure Internal (1)
リソース グループ* 20210611-handa (2)
新規作成

インスタンスの詳細
この名前空間に必要な設定を入力します。

名前空間の名前* sb-handa-handson (3).servicebus.windows.net

場所* 東日本 (4)

価格レベル (価格の詳細を表示) * Basic (5)

確認および作成 < 前へ 次へ: タグ >

以下の項目を、適宜設定ください。
※必要に応じて、設定内容をメモお願いします

- ① サブスクリプション :
- ② リソースグループ :
- ③ 名前空間の名前 :
- ④ 場所 :
- ⑤ 価格 :

「確認および作成」をクリック後、「作成」ボタンにて
デプロイ開始

キューの作成

Service Busの画面にて、以下の順でボタンを押下します。

- ① エンティティにて[キュー]を選択
- ② [+キュー] を選択

The screenshot shows the Azure Service Bus Management Portal. On the left, there's a sidebar with options like '検索 (Ctrl+ /)', '問題の診断と解決', '設定' (with '共有アクセス ポリシー', 'geo リカバリー', 'Premium へ移行', '暗号化', 'プロパティ', 'ロック'), 'エンティティ' (with 'キュー' highlighted by a red box and circled with a red number 1), and 'トピック'. The main area is titled 'sb-handa-handson | キュー' and shows a table with one row: '結果なし。'. A large red box highlights the '+ キュー' button in the top right, which is circled with a red number 2.

The screenshot shows the 'Queue Creation' dialog box. It has fields for '名前' (alert-queue), '最大キュー サイズ' (1 GB), '最大配信数' (10), and time-related settings ('メッセージの Time to Live' and 'ロツク期間'). There are also several checkboxes at the bottom. A large grey arrow points from the '+ キュー' button on the left screen to this dialog box.

名前 *	alert-queue
最大キュー サイズ	1 GB
最大配信数 *	10
メッセージの Time to Live	日: 14 時間: 0 分: 0 秒: 0
ロツク期間	日: 0 時間: 0 分: 0 秒: 30
<input type="checkbox"/> アイドル キューの自動削除を有効にする	
<input type="checkbox"/> 重複データ検出を有効にする	
<input type="checkbox"/> メッセージの期限切れによる配信不能を有効にする	
<input type="checkbox"/> パーティション分割を有効にする	
<input type="button" value="作成"/>	

以下の項目を適宜設定し[作成]をクリック
✓ 名前 : (例) alert-queue
✓ その他はデフォルト設定

ポリシーの作成

Service Busの画面にて、以下の順でボタンを押下します。

- ① エンティティにて[キュー]を選択し、作成したキューを選択
- ② 設定にて[共有アクセス ポリシー]を選択し、[+追加]を選択

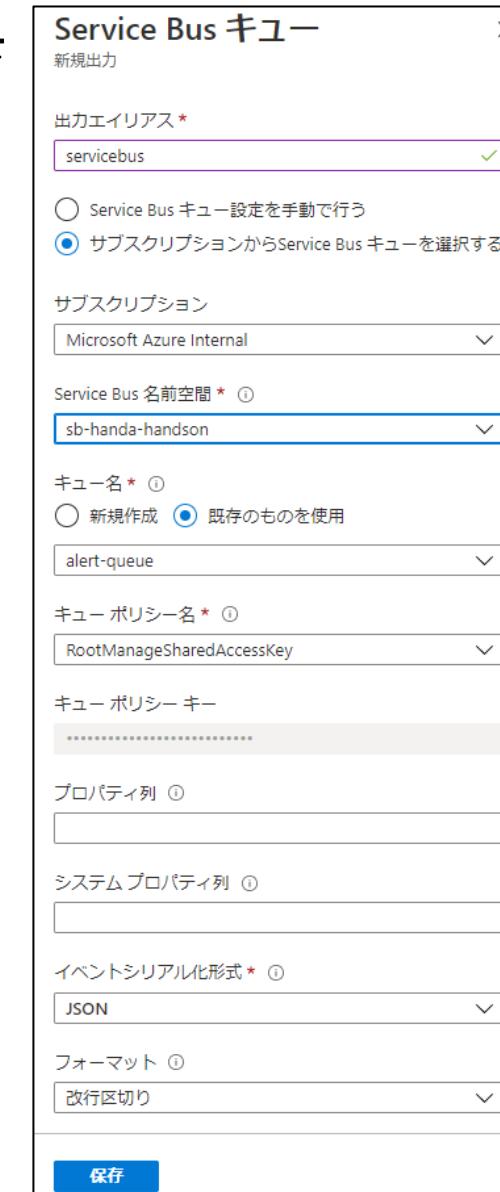
以下の項目を適宜設定し[作成]をクリック
✓ ポリシーネ名：（例）alert-policy
✓ [管理] をオンにする



Stream Analytics の出力設定 (Service Bus)

Stream Analyticsの画面にて、以下の順でボタンを押下

- ① ジョブトポロジにて[出力]を選択
- ② [追加] → [Service Bus キュー] を選択



以下の項目を適宜設定し[保存]をクリック

- ✓ 出力エイリアス : (例) **servicebus**
- ✓ Service Bus 名前空間 : 以前の手順で作成した名前空間を選択
- ✓ キュー名 : [既存のものを使用] を選択し、以前の手順で作成したキューを選択
- ✓ その他はデフォルト設定

Stream Analytics のクエリ設定

Stream Analyticsのクエリ画面にて、以下の通りService Busへの出力設定を追加
その後、クエリを保存し、Stream Analyticsを起動

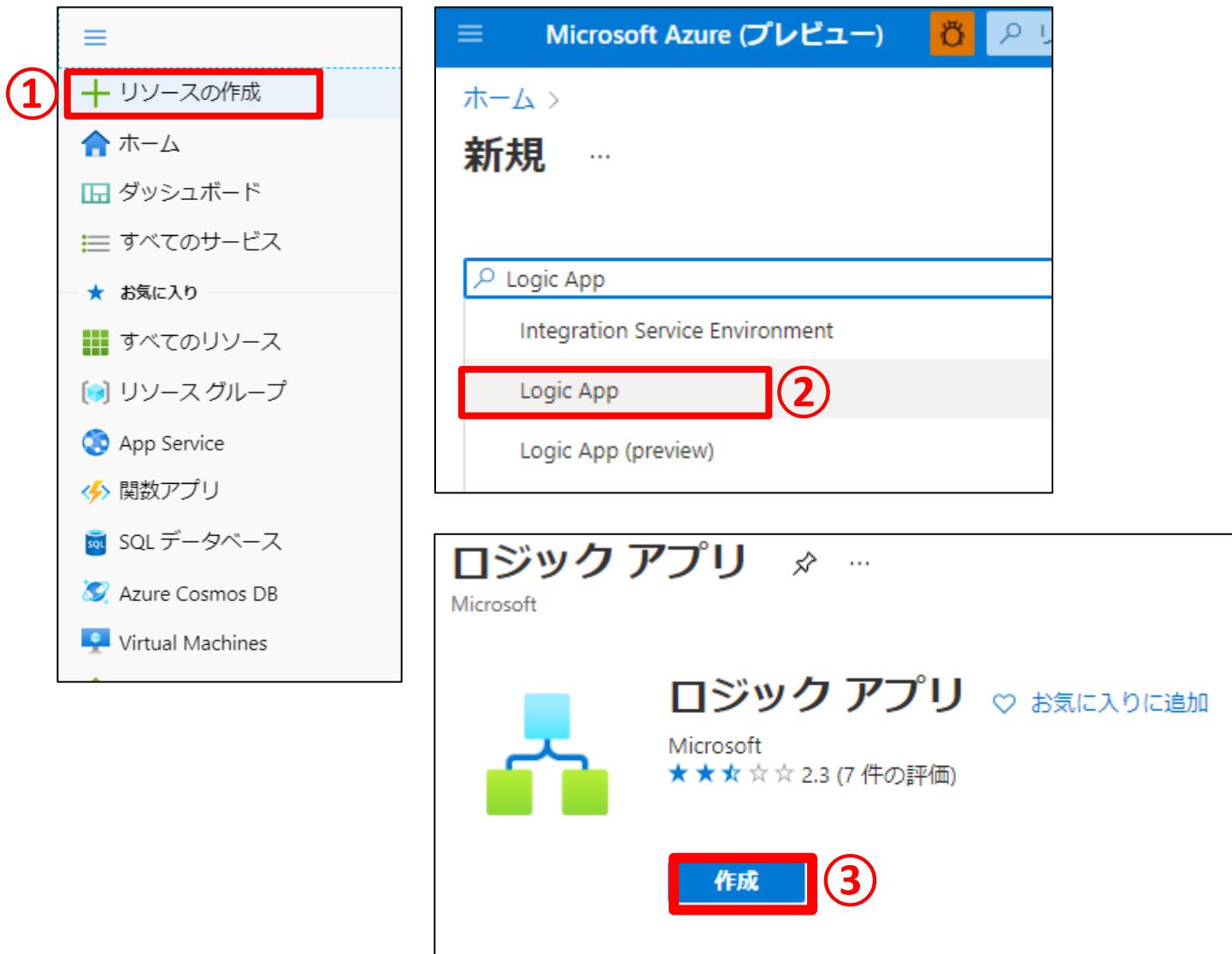
The screenshot shows the Stream Analytics Query Editor interface. On the left, there are two sections: 'Input (1)' containing 'iohub' and 'Output (2)' containing 'adls' and 'servicebus'. The main area displays a T-SQL query:

```
/* DataLake Storage への出力 */
SELECT *
INTO [adls]
FROM [iohub]

/* Service Bus への出力 */
SELECT
    Start_time,
    End_time,
    -- 開始時刻と終了時刻の差分を作業時間(Work_time)として算出
    DATEDIFF(
        second,
        CAST(SUBSTRING(Start_time ,1 ,4)+SUBSTRING(Start_time ,6 ,2)+SUBSTRING(Start_time ,9 ,2)+SUBSTRING(Start_time ,11 ,17) AS DATETIME),
        CAST(SUBSTRING(End_time ,1 ,4)+SUBSTRING(End_time ,6 ,2)+SUBSTRING(End_time ,9 ,2)+SUBSTRING(End_time ,11 ,17) AS DATETIME)
    ) as Work_time,
    Work,
    Line_num,
    Lot_num
INTO [servicebus]
FROM [iohub]
```

The 'Output (2)' section is expanded to show 'adls' and 'servicebus'. The 'servicebus' entry is highlighted with a blue selection bar.

Logic App の作成



1. Azure Portalへのアクセス
<https://ms.portal.azure.com>

2. 以下の順にメニューを押下

- ① [+ リソースの作成]
- ② [Logic App] を検索
- ③ [作成] を選択

Logic App の各種設定

ロジック アプリの作成 ...

基本 タグ 確認および作成

ロジック アプリを作成して、リソースを簡単に管理、デプロイ、共有するためにワークフローを論理ユニットとしてグループ化できるようにします。ワークフローを使用すると、ビジネスに不可欠なアプリとサービスを Azure Logic Apps に接続し、コードを 1 行も記述せずにワークフローを自動化できます。

プロジェクトの詳細

デプロイされているリソースとコストを管理するサブスクリプションを選択します。フォルダーのようなリソースグループを使用して、すべてのリソースを整理し、管理します。

サブスクリプション* ①

Microsoft Azure Internal ①

リソースグループ* ②

20210611-handa ②

新規作成

インスタンスの詳細

タイプ* ③

消費 Standard

③ 従来の従量課金の作成エクスペリエンスをご希望ですか? [こちらをクリック](#)

ロジック アプリ名* ④

logic-handa-handson ④

地域* ⑤

Japan East ⑤

ログ分析を有効化* ⑥

はい いいえ

確認および作成

< 前へ

次: タグ >

以下の項目を、適宜設定ください。

*必要に応じて、設定内容をメモお願いします

① サブスクリプション :

② リソースグループ :

③ タイプ : [消費] を選択

④ ロジック アプリ名 :

⑤ 地域 :

「確認および作成」をクリック後、「作成」ボタンにて
デプロイ開始

Logic App のトリガーの構成

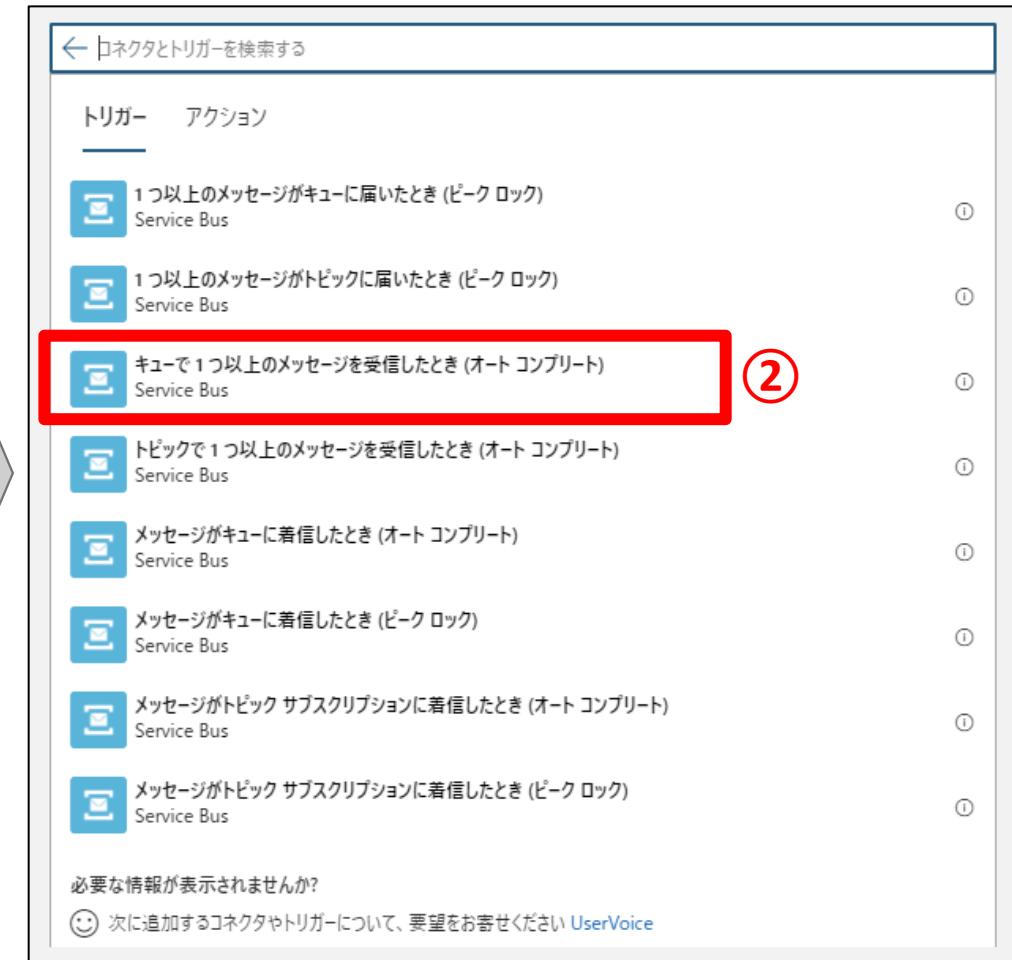
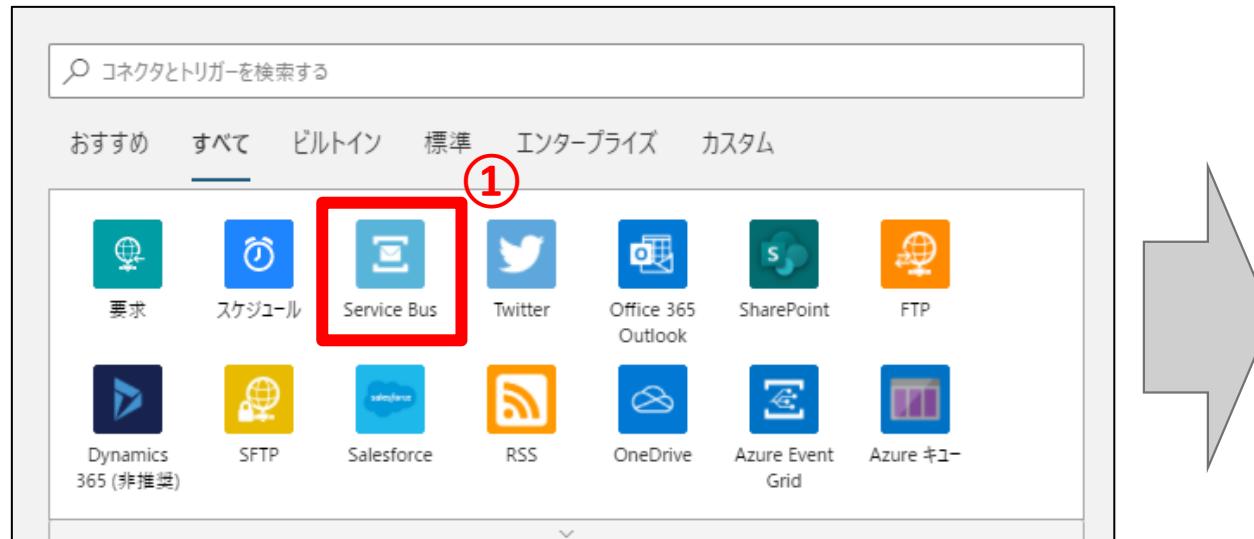
Logic Appの画面にて、以下の順でボタンを押下します。

- ① 開発ツールにて[ロジック アプリ デザイナー]を選択
- ② テンプレートまで下へスクロールし、[空のロジック アプリ]を選択



Logic App のトリガーの構成

- ① [すべて] タブを選択して、[Service Bus]を選択
- ② [トリガー]で、[キューで1つ以上のメッセージを受信したとき (オート コンプリート)]を選択



Logic App のトリガーの構成

- ① [キュー名] で、ドロップダウンから先の手順で作成したキューを選択
- ② [項目を確認する頻度] を3秒に設定
- ③ [+新しいステップ] を選択



Logic App のアクションの構成

- ① [すべて] タブを選択して、検索ボックスに[制御]と入力し、[制御]を選択
- ② [アクション]で、[条件]を選択



Logic App のアクションの構成

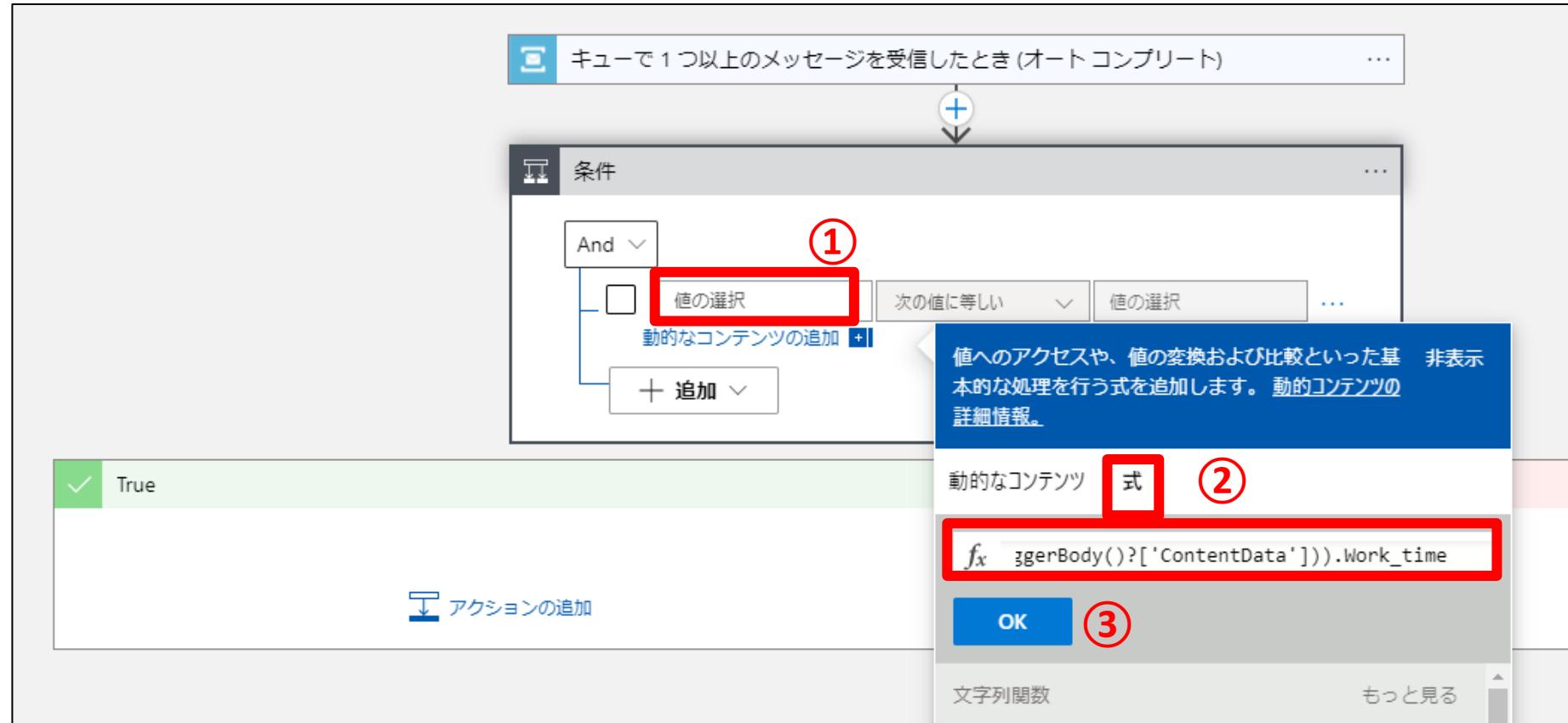
① [条件] にて [値の選択] を選択

② [式] タブを選択して、以下式を入力

```
json(base64ToString(triggerBody()?'ContentData')).Work_time
```

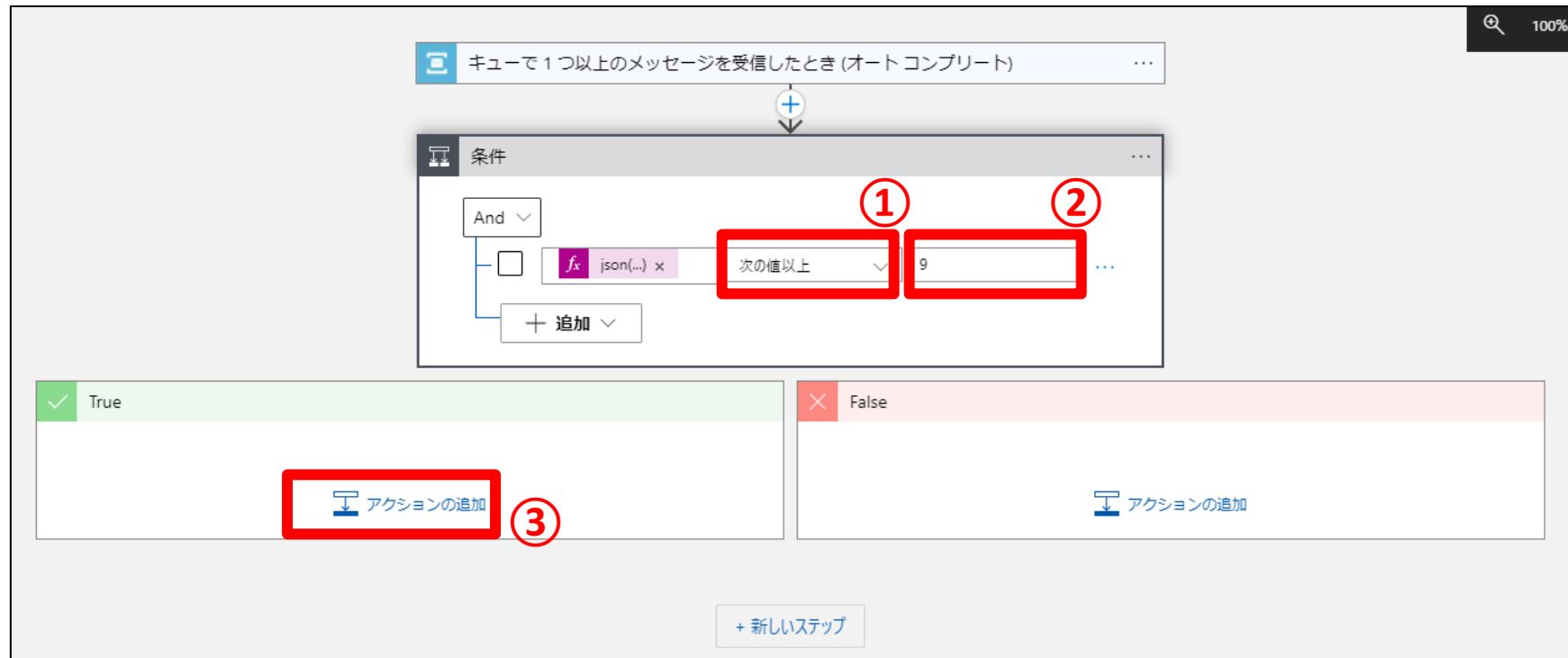
Stream Analyticsで出力した Work_time
(作業開始時刻と作業終了時刻の差分
から算出した作業時間) を取得

③ [OK] を選択



Logic App のアクションの構成

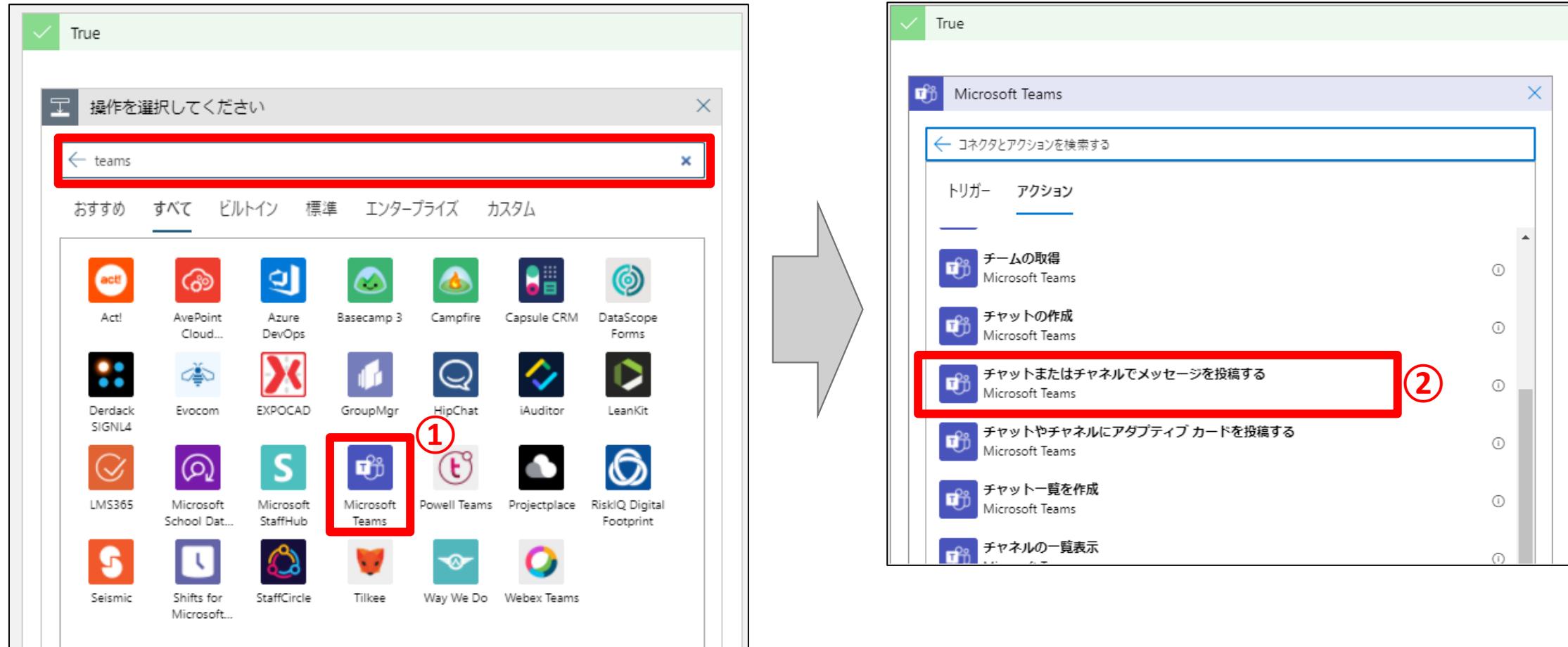
- ① ドロップダウンより、[次の値以上] を選択
- ② しきい値に [9] を入力 ※アラート上げる際のしきい値をここで指定しています。
- ③ True 内の [アクションの追加] を選択



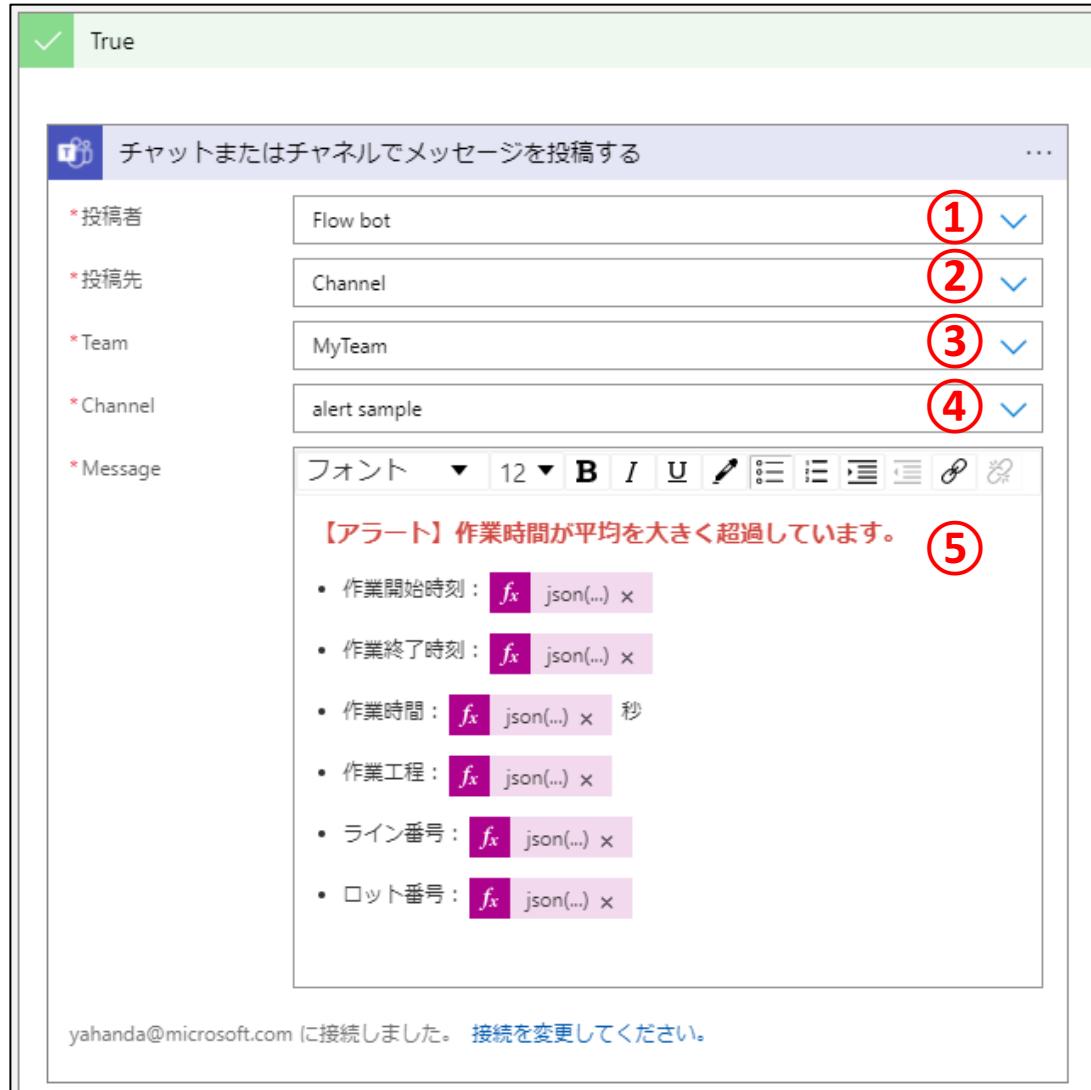
Logic App のアクションの構成

- ① [すべて] タブを選択して、検索ボックスに[teams]と入力し、[Microsoft Teams]を選択
- ② [アクション]で、[チャットまたはチャネルでメッセージを投稿する]を選択

※サインイン画面が出たら、ご自身のTeams IDにてサインインします



Logic App のアクションの構成

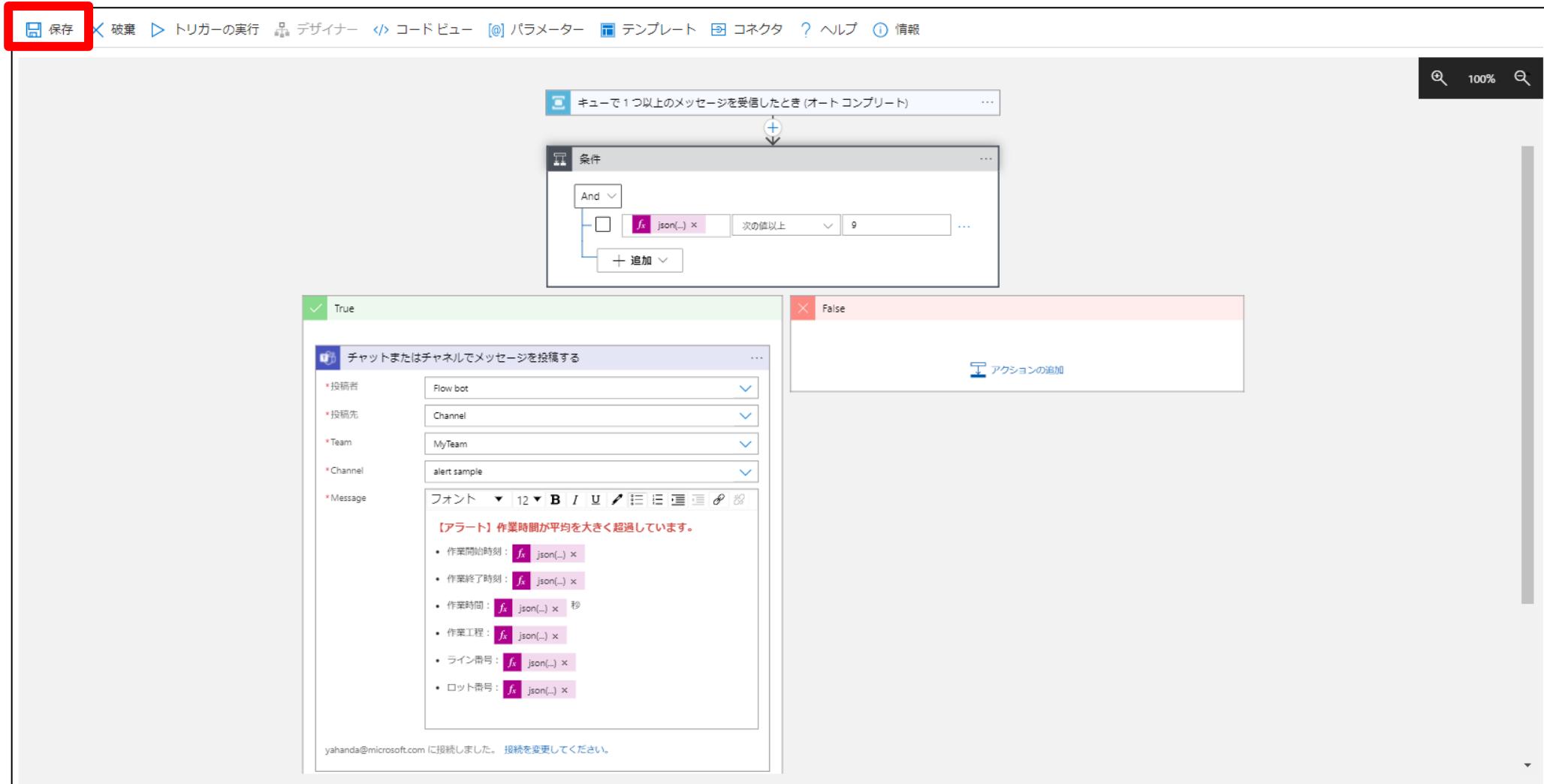


以下の項目を、適宜設定ください。

- ① 投稿者 : [Flow bot] を選択
- ② 投稿先 : [Channel] を選択
- ③ Team :
- ④ Channel :
- ⑤ Message :
※[動的なコンテンツの追加] > [内容] より Stream Analytics から送信されたメッセージのコンテンツが追加できます

Logic App のアクションの構成

[保存]を選択して、変更内容を保存



Teams アラートの確認

- Logic App にて指定した条件で、Teams アラートが表示されることを確認します

The screenshot shows a Microsoft Teams channel named "alert sample". It contains two messages from a user named Yasuhiro Handa via Flow at 8:24 PM. Both messages are alerts about work duration exceeding average, with detailed log entries.

Message 1:

Yasuhiro Handa via Flow 8:24 PM
【アラート】作業時間が平均を大きく超過しています。

- ・作業開始時刻：2021-08-18 20:24:16.817414
- ・作業終了時刻：2021-08-18 20:24:25.829300
- ・作業時間：9 秒
- ・作業工程：WORK07
- ・ライン番号：LINE001
- ・ロット番号：LOT021

Message 2:

Yasuhiro Handa via Flow 8:24 PM
【アラート】作業時間が平均を大きく超過しています。

- ・作業開始時刻：2021-08-18 20:24:25.844981
- ・作業終了時刻：2021-08-18 20:24:35.849044
- ・作業時間：10 秒
- ・作業工程：WORK08
- ・ライン番号：LINE001
- ・ロット番号：LOT021

A callout bubble points from the second message to a note: "作業時間が9秒以上の場合、アラートが通知される" (An alert is notified when the work duration exceeds 9 seconds).

