５．まとめ

本章ではこれまで紹介した個別の技術をまとめた IoT システムをいくつか試します。

5.1　世界の温度計

(1)BBG にデバイスを作る

前章で試した Node-RED によるデバイスを BBG で構築します。コネクションストリングはデバイス側の接続文字列を使います。DeviceExplorer の Management タグから得ます。

○温度をアップしてみる

BBG のデバイスからは温度をアップできるようにします。全体のフローおよび実行の様子は図 5.1 のようになります。アナログインノードは数値が出力されるのでファンクションノードを途中に配置して文字列に変換し Azure ノードに入力します。クラウド側は DeviceExplorer を使って送られてきている状態をチェックします。

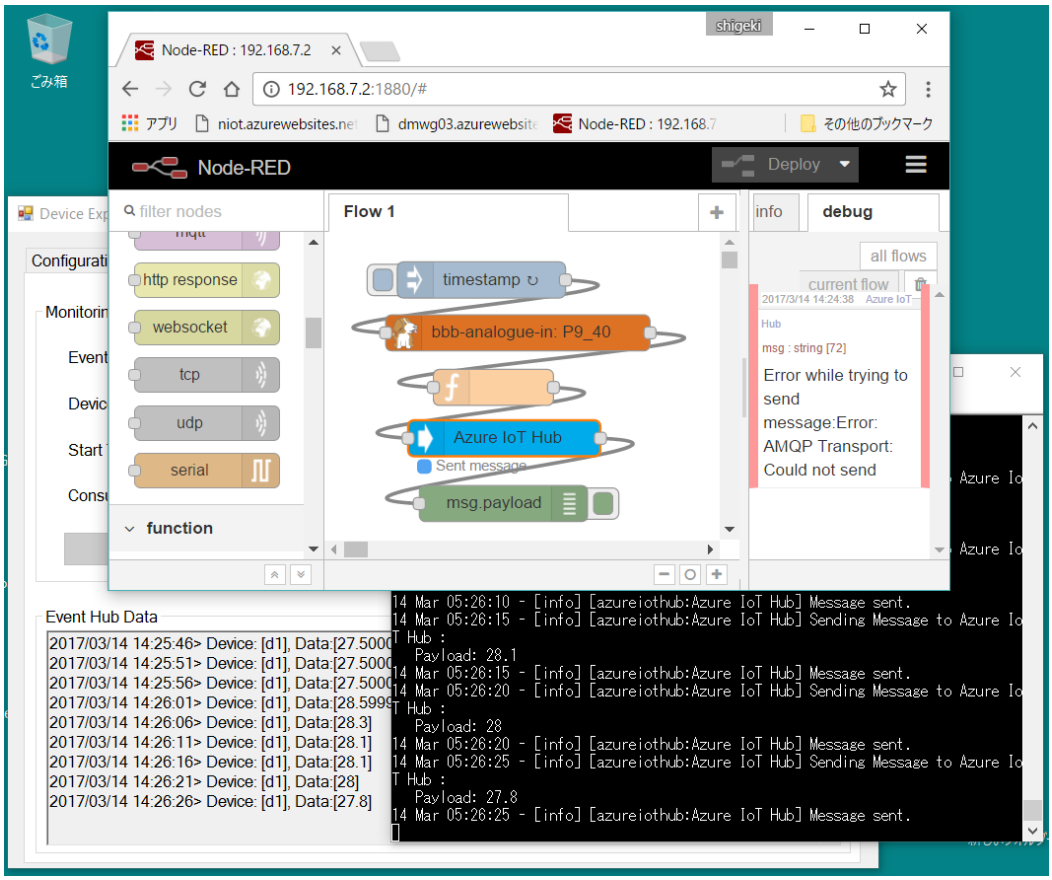

図 5.1　温度を IoTHub にアップするフローと実行の様子

リスト 5.1　数値を文字列に変換するファンクションノード

```
msg.payload = msg.payload.toString();
return msg;
```

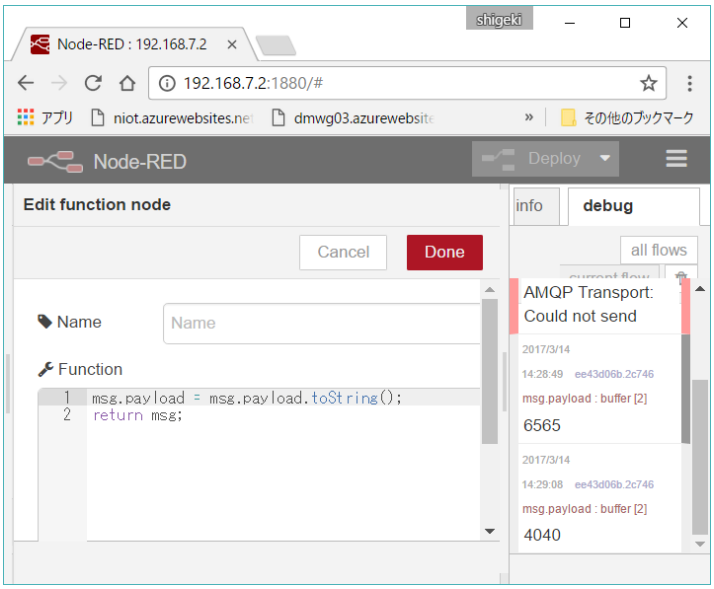

図 5.2　数値を文字列に変換するファンクションノード

(2) 温度表示システム

続いてデバイスで温度計測を行い、そのデータをクラウドにアップして、ブラウザで見ることのできるシステムを試します。これまでは DeviceExplorer で確認していた IoTHub に届いているデータを Azure 上のアプリケーションから読み出し、それを web ページへ転送することで実現します。

○ 温度データをブラウザから見えるようにする　～dashboard を使う～

デバイスからの温度データの受信を行い、任意のブラウザからアクセスできるように WebApps 上の Node-RED を使用し dashboard のグラフ機能で温度データ表示ページの配信を行います。全体は図 5.3 のようになります。

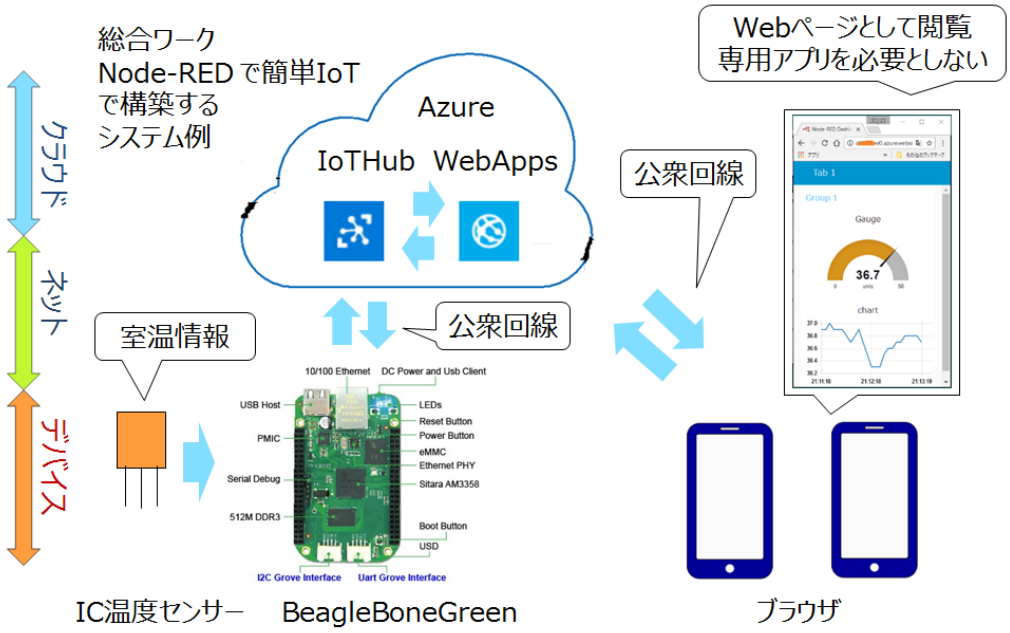

図 5.3

実行の様子を図 5.4 に示します。

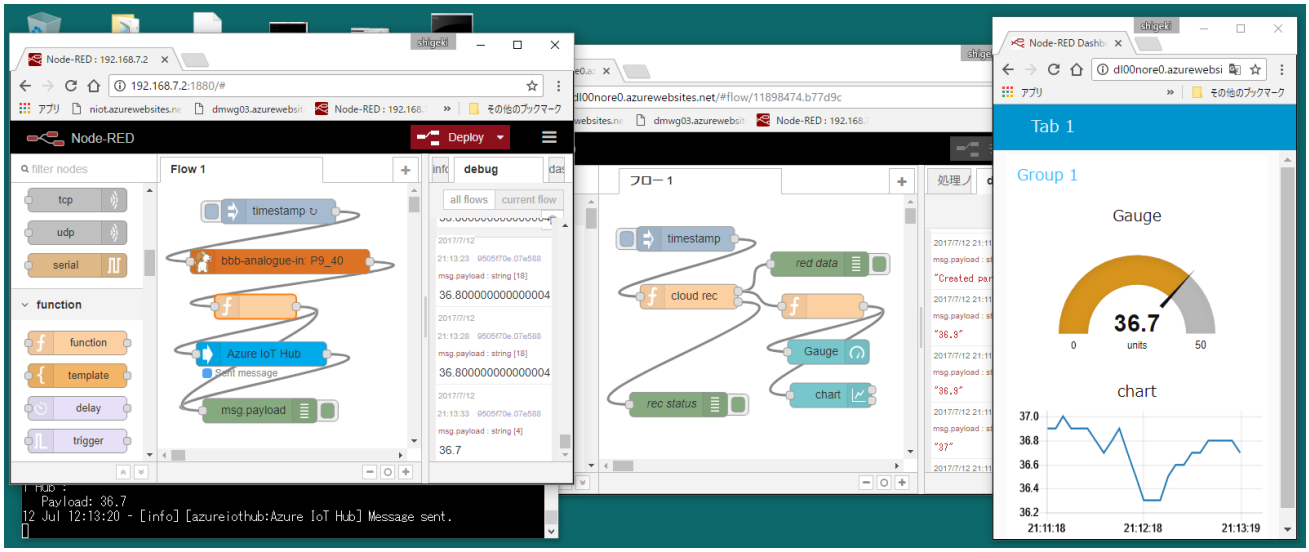

図 5.4　実行の様子

Azure 側のフローのファンクションノードには IoTHub からの受信データの取り出しと文字列を数値に直すコードを用意します。

リスト 5.3　数値を文字列に変換するファンクションノード

```
msg.payload = parseFloat(msg.payload);
return msg;
```

リスト 5.2　ファンクションノード cloud rec のリスト(fn401a)

```
// evhbClient
var connectionString=
'HostName=doilab.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=d0Ixxxx';

var prec = context.get('prec') || 0;
var printError = function (err) {
  console.log(err.message);
};

var printMessage = function (message) {
  console.log('Message received: ');
  console.log(JSON.stringify(message.body));
  console.log('');
  msg.payload = JSON.stringify(message.body);
  node.send( [msg, null] );
//  return [msg, null] ;
};

if( prec ){
    msg.payload ='receiver aleady open';
    return [null,msg];
}
else{
    var client = context.global.evhbClient.fromConnectionString(connectionString);
    client.open()
    .then(client.getPartitionIds.bind(client))
    .then(function (partitionIds) {
        return partitionIds.map(function (partitionId) {
            return client.createReceiver('$Default', partitionId,
                { 'startAfterTime' : Date.now()})
                .then(function(receiver) {
                    console.log('Created partition receiver: ' + partitionId);
                    msg.payload = 'Created partition receiver: ' + partitionId;
                    node.send([null,msg]);
                    prec = 1;
                    context.set('prec',prec);
                    receiver.on('errorReceived', printError);
                    receiver.on('message', printMessage);
                });
            });
        })
    .catch(printError);
}
```

〇下り線も追加してみる

さらに IoTHub からの下り線で LED を制御してみます。IoTHub からの送信データにより LED をオンオフするデータを作ります。そのためにもう 1 つファンクションノードを用意します。

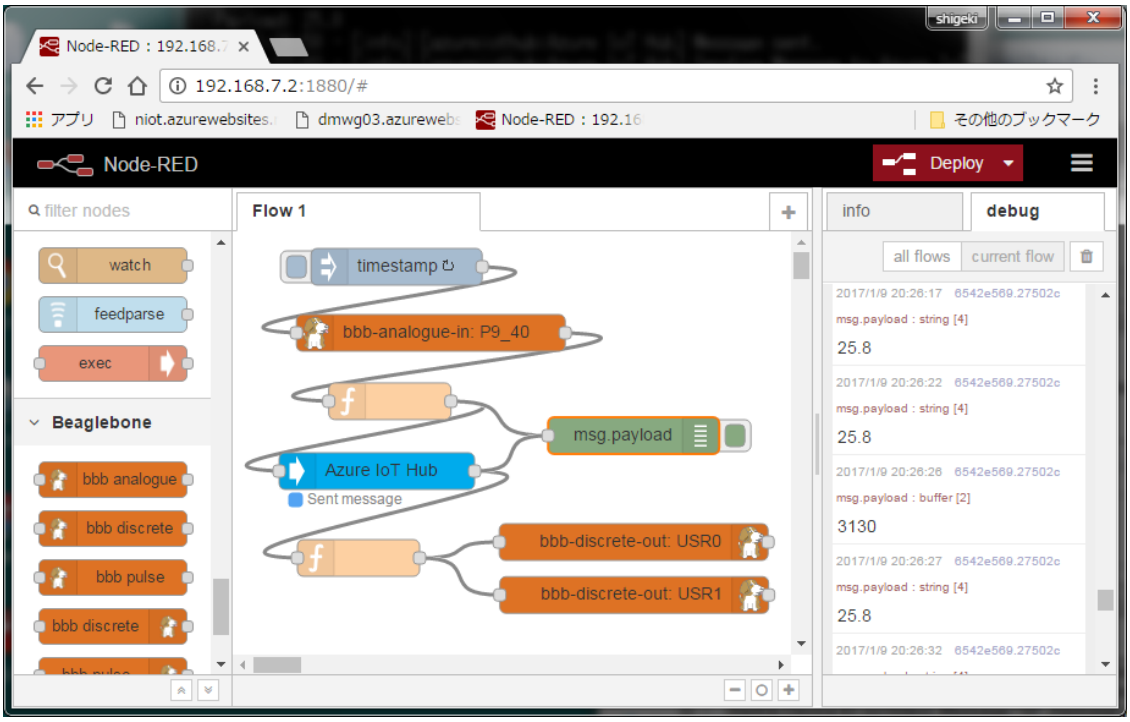

図 5.5

〇LED 制御用ファンクションノードの設定　　～受信データの最初の値を'0'と比較する。

リスト 5.4

```
if(msg.payload[0]==0x31)
    msg.payloag = 1;
else
    msg.payload = 0;
return msg;
```

5.2　TableStorage を使う

クラウドでデータを保存する場合、ボードマイコンなどの場合のメモリ、PC の場合のファイルに相当するものは、メモリやファイルでなく、データベースサービスを利用することになります。このデータベースサービスにはリレーショナルデータベースとして利用できるもの、スタイルシートのように利用できるもの、データの塊として利用するものなど、いくつかの種類が用意されています。

IoTHub などからのストリームとして流し込まれるデータにはスタイルシートのデータのように利用できる TableStorage を使うのが適当のようです。TableStorage に関連するデータの流れを図 5.6 に示します。TableStorage へは StorageExplorer などのツールや、API を用いてアクセスでき、データの出し入れができます。また、StreameAnalytics を経由して IoTHub との間でデータの転送ができます。

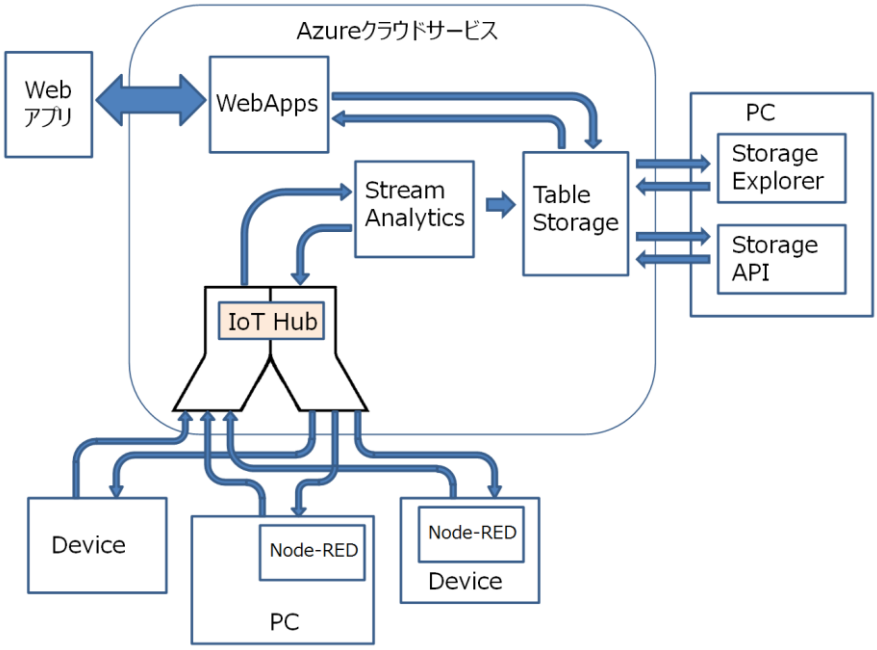

図 5.6　Table Storage とその周辺

(1) TableStorage を使うための準備

○Storage Explorer の準備

Storage Explorer をインストールしておきます。

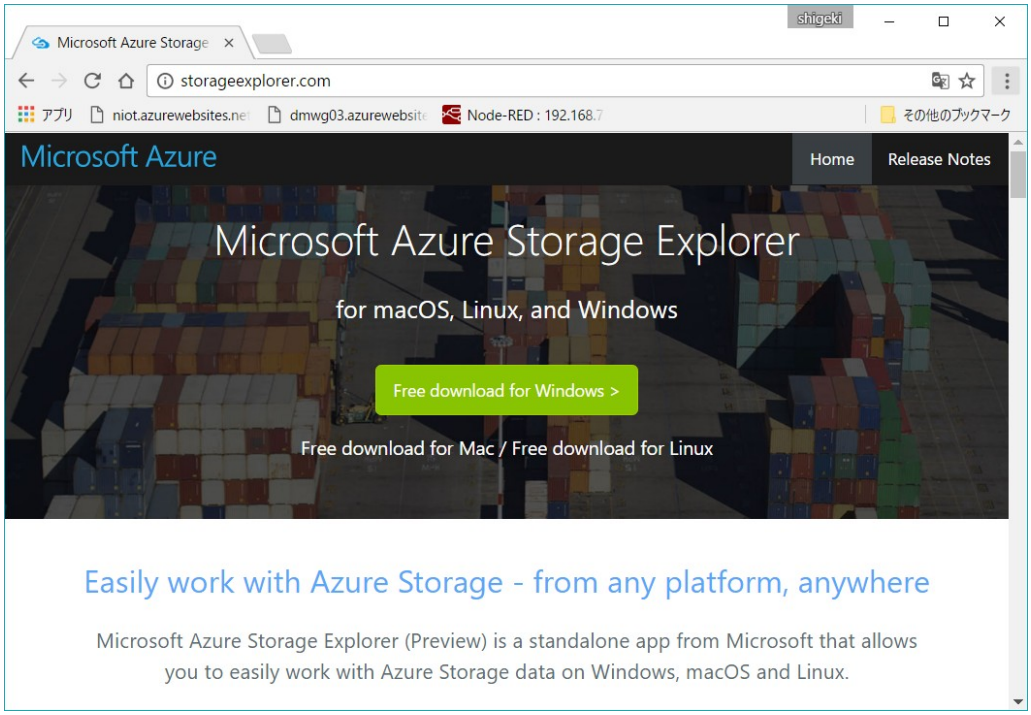

図 5.7　Storage Explorer のページ

○Stream Analytics の準備

新しく Stream Analytics サービスを作成します。StreamAnalytics の名前になる「ジョブ名」を入力し、リソースグループを選択し、「作成」をクリックします。

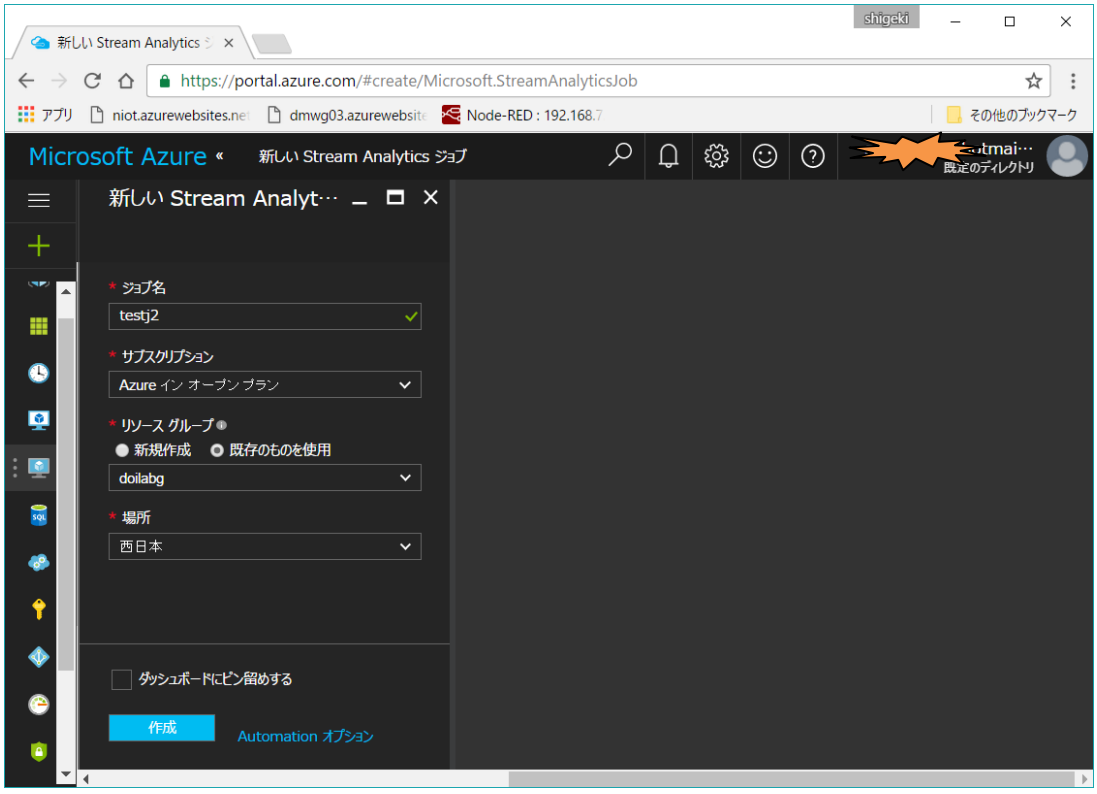

図 5.8　StreamAnalytics の作成

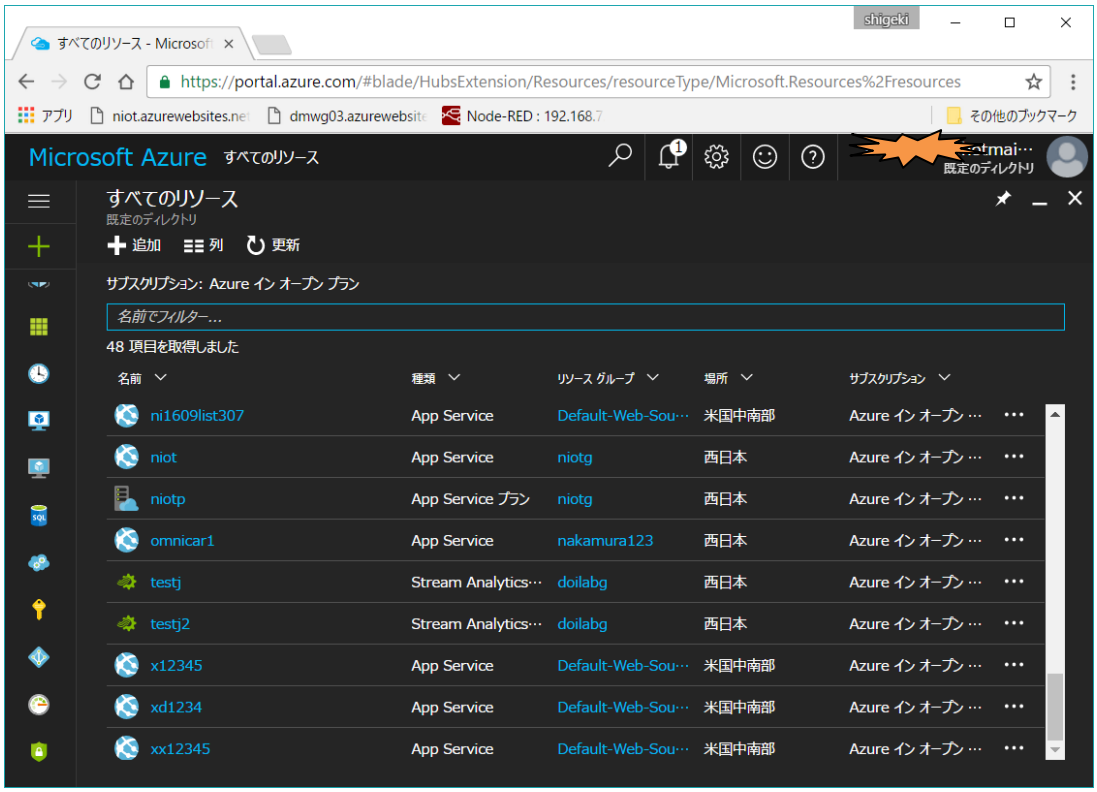

図 5.9　作成された StreamAnalytics

続いて、StreamAnalytics の設定、「入力」「クエリ」「出力」の３つを行います

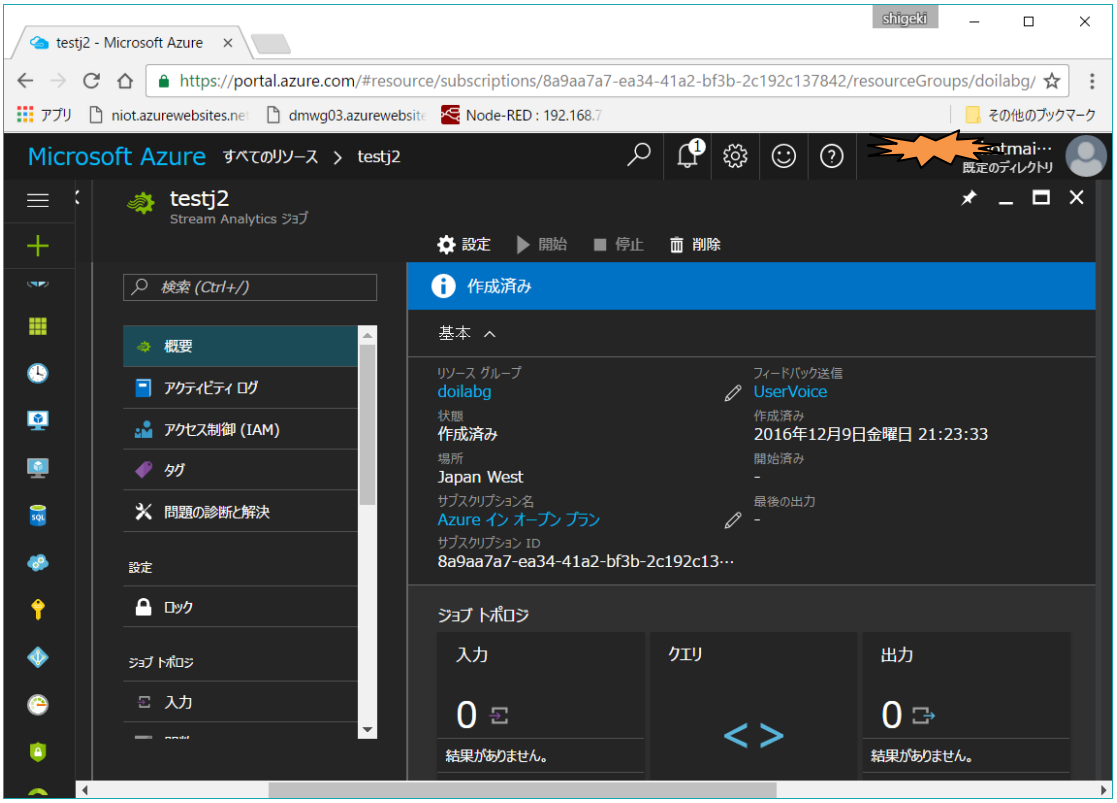

図 5.10　「入力」「クエリ」「出力」の３つを設定する

図 5.10 の「入力」をクリックし、入力画面で「追加」をクリックすると図 5.11 の新しい入力画面となります。「入力のエイリアス」で入力に名前を付けます。この名前はあとの「クエリ」で使います。続いて「ソース」項目で「IoTHub」を選択します。すると IoTHub 用のパラメータが自動的に入ります。最後に「作成」　をクリックします。

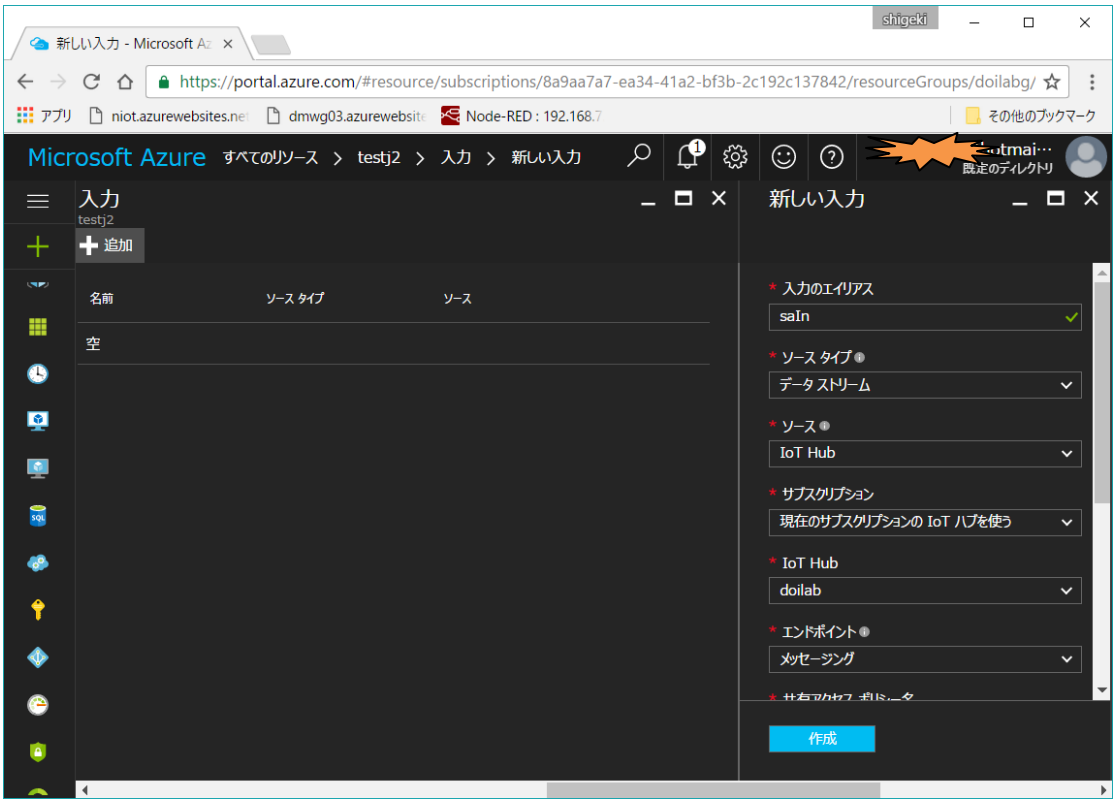

図 5.11　「入力」には IoTHub を設定する

続いて「クエリ」を設定します。「クエリ」には「入力のエイリアス」と「出力のエイリアス」それぞれの名前を記述します。「クエリ」はその名のとおり、さまざまな「問合せ処理」を SQL に似たクエリ言語で書くことができますが、ここではすべてを素通りさせる　* を記述します。

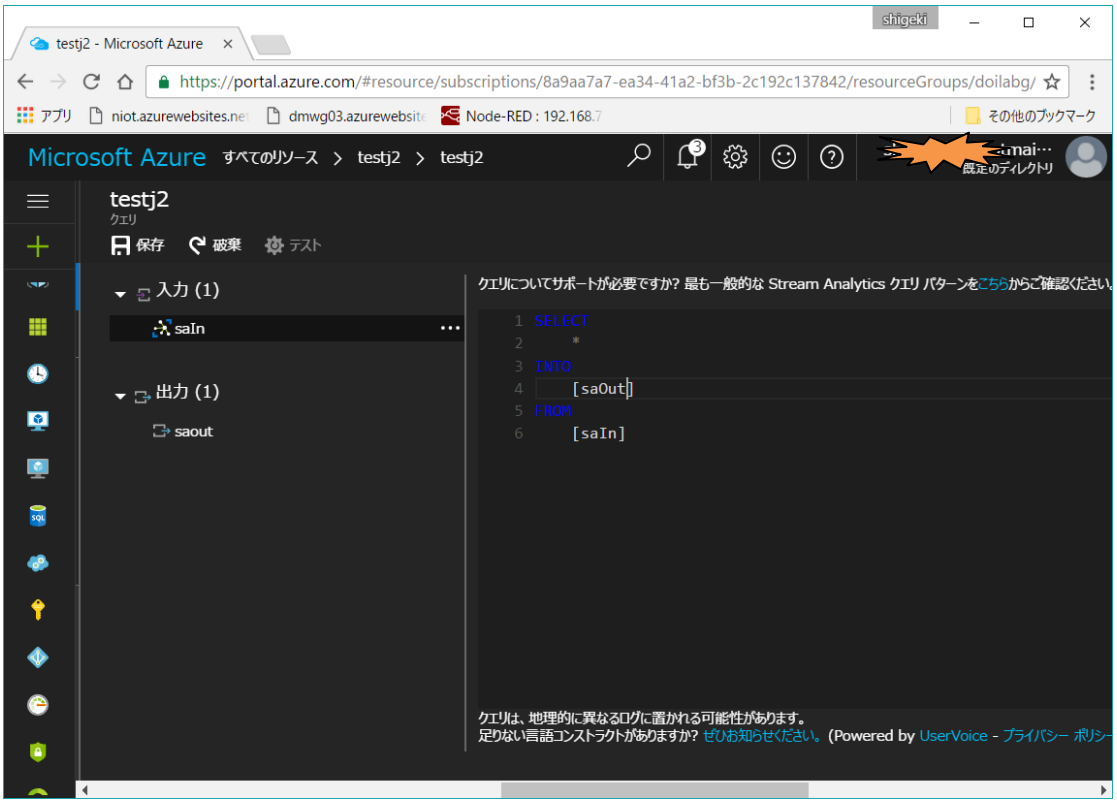

図 5.12　「クエリ」は「素通り」を設定します

○ストレージアカウントの準備

のこりの「出力」の設定でストレージアカウントも生成します。「テーブル名」「パーティションキー」「行キー」の名前を設定します。図 5.13 に示すように TableStorage は２つのキーでインデックスされます。StreamAnalytics でストレージアカウントを出力に設定すると、IoTHub に届いたデータのうち、このキーのあるもののみを対応するテーブルストレージに送ります。

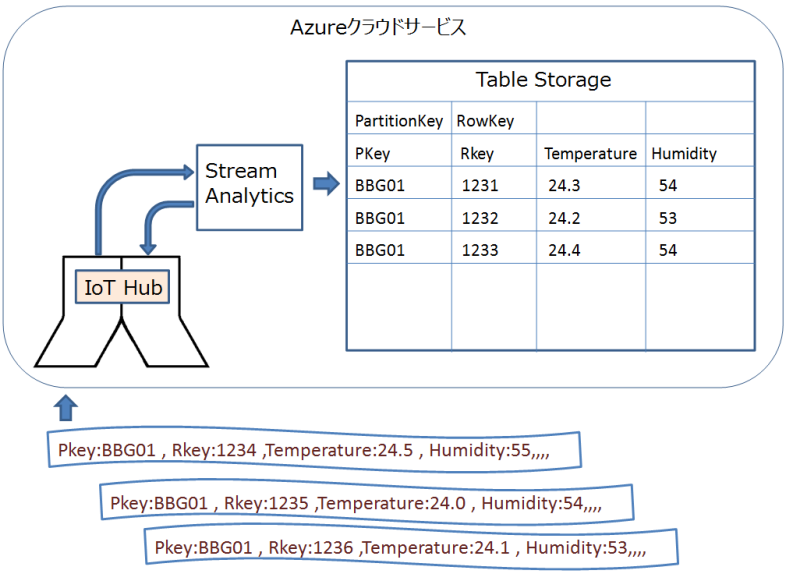

図 5.13　受信データと TableStorage

「出力」も図 5.14 に示すように「入力」の設定と同様に設定を行います。まず、「出力エイリアス」の名前を入力します。これは「クエリ」で記述したものと同じにします。続いて、「シンク」項目で「テーブルストレージ」を選択します。すると項目が「テーブルストレージ」用に自動的に切り替わります。「ストレージアカウント」項目で「新しいストレージアカウントの作成」を選択します。「ストレージアカウント」に名前を入力します。この名前でストレージアカウントが作成されます。URL のようにユニークであることが必要です。更に「テーブル名」にストレージアカウント内に作られるテーブルストレージの名前を入力します。

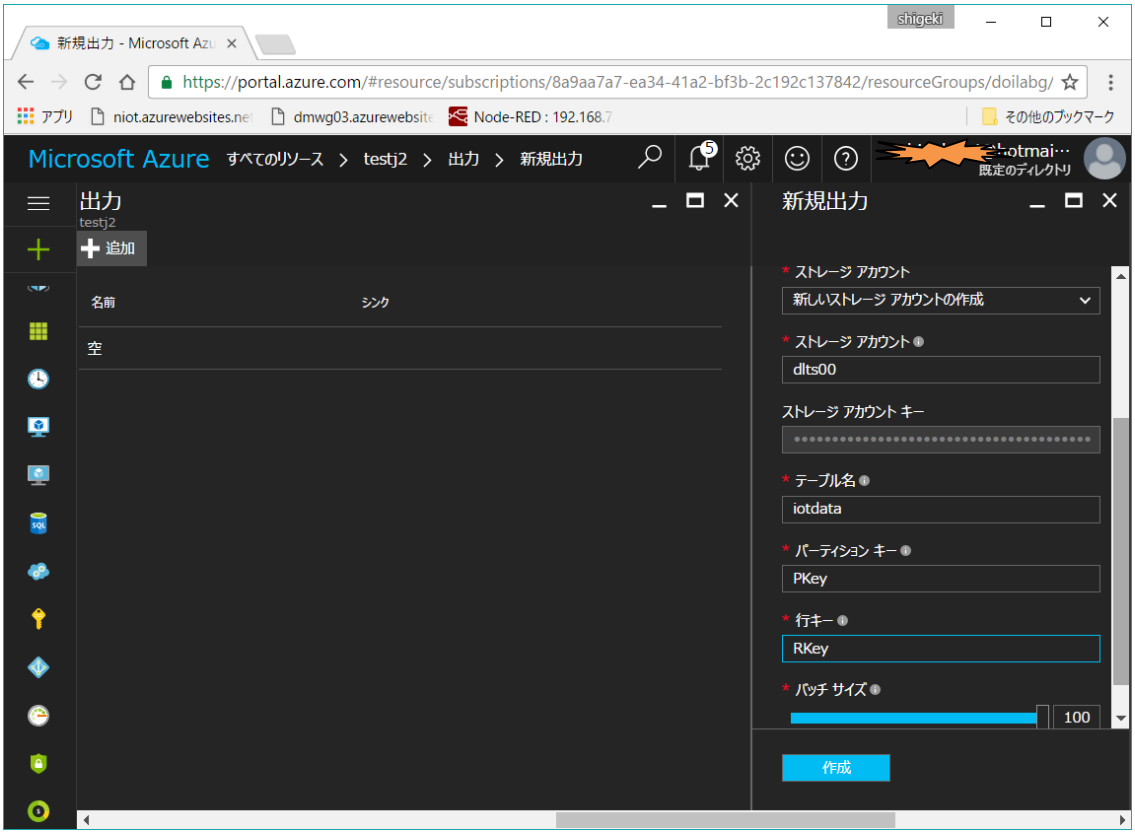

図 5.14

作成されたストレージを確認します。

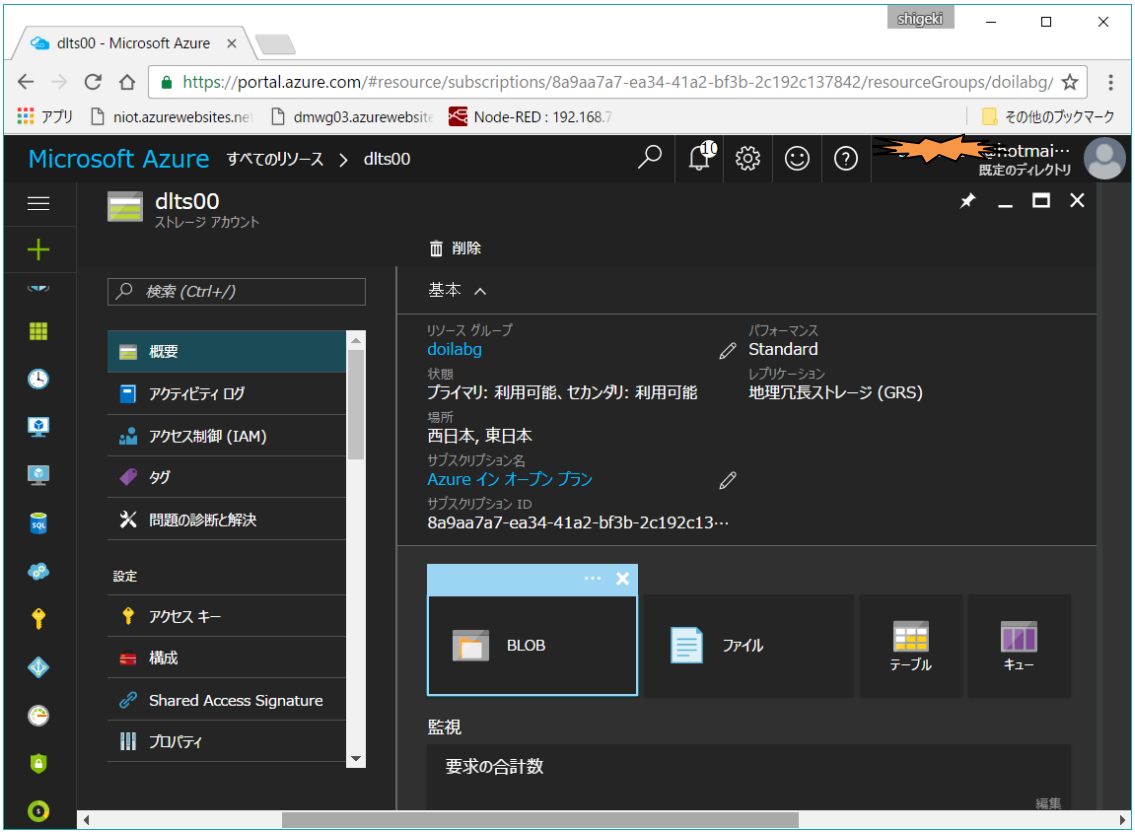

図 5.15　作成されたストレージ

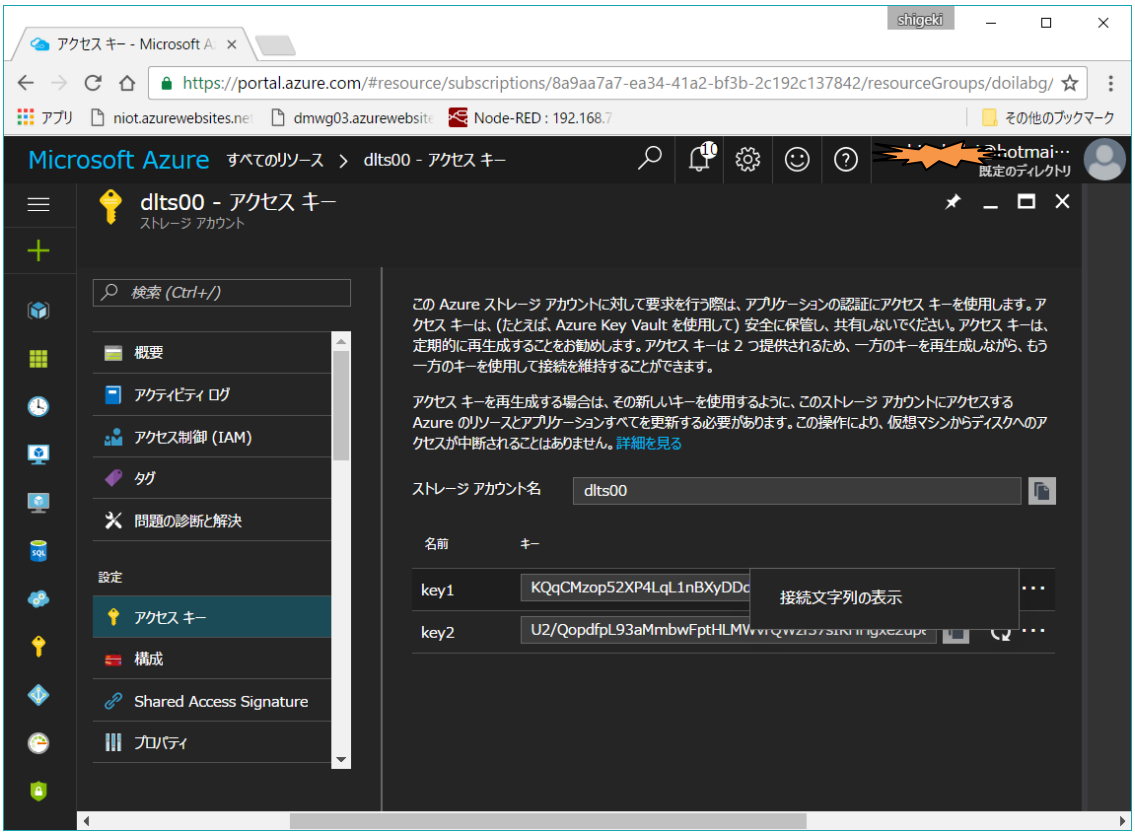

図 5.16

StorageExplorer からアクセスしてみます。人の上半身のようなアイコンから Azure にログインすると、そのアカウントのストレージが表示されます。その様子を図 5.15 に示します。新しく作られたストレージアカウントとその中にテーブルがあるのが分ります。

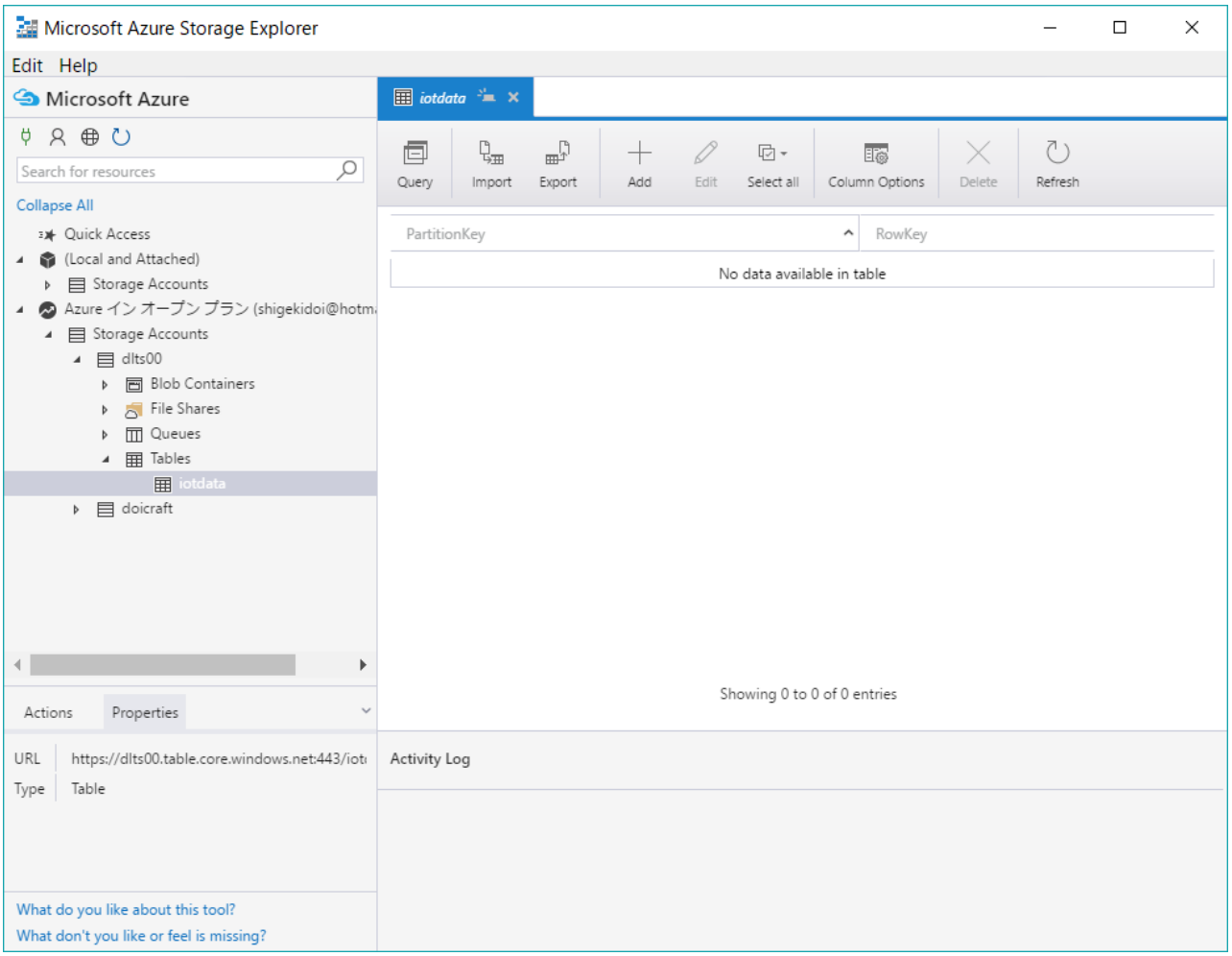

図 5.17

(2)　温度データをアップしてみる

続いて作成した TableStrage にデータを流し込んでみます。

○デバイスの準備

BBG による温度計測デバイスを準備します。Node-RED を使ってデバイスをつくります。データには「パーティションキー」「行キー」を含めます。デバイスを動かして、ストレージの状態を　StorageExplorer で確かめます。

○実行

あらかじめ StreamAnalytics の「開始」ボダンでクラウド側の動作を開始しておきます。

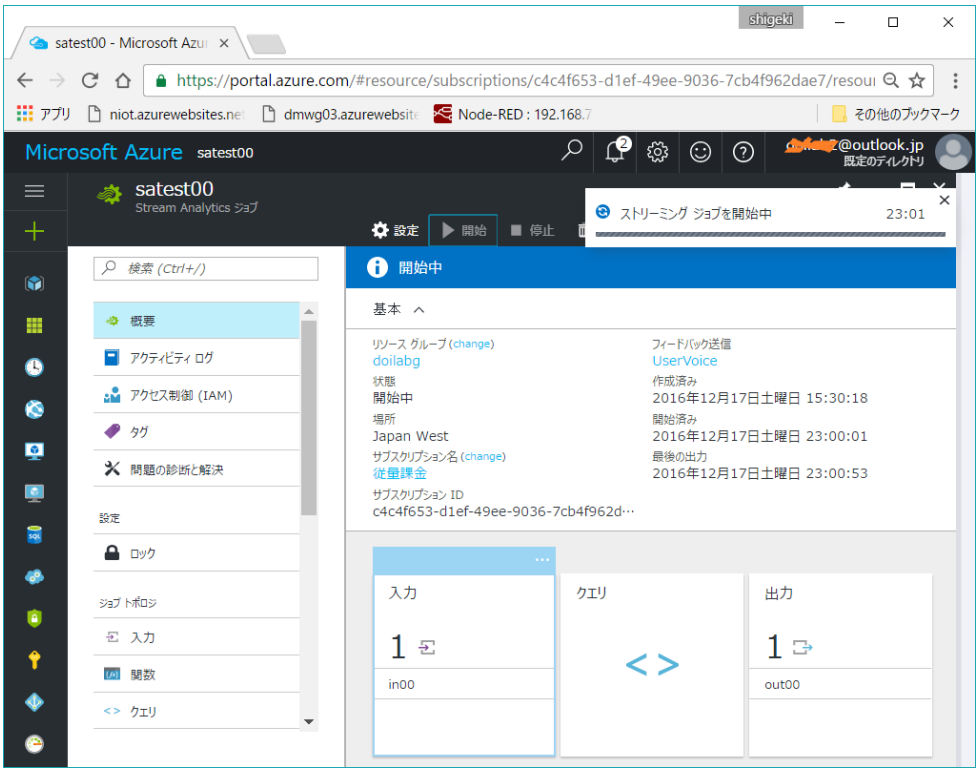

図 5.18

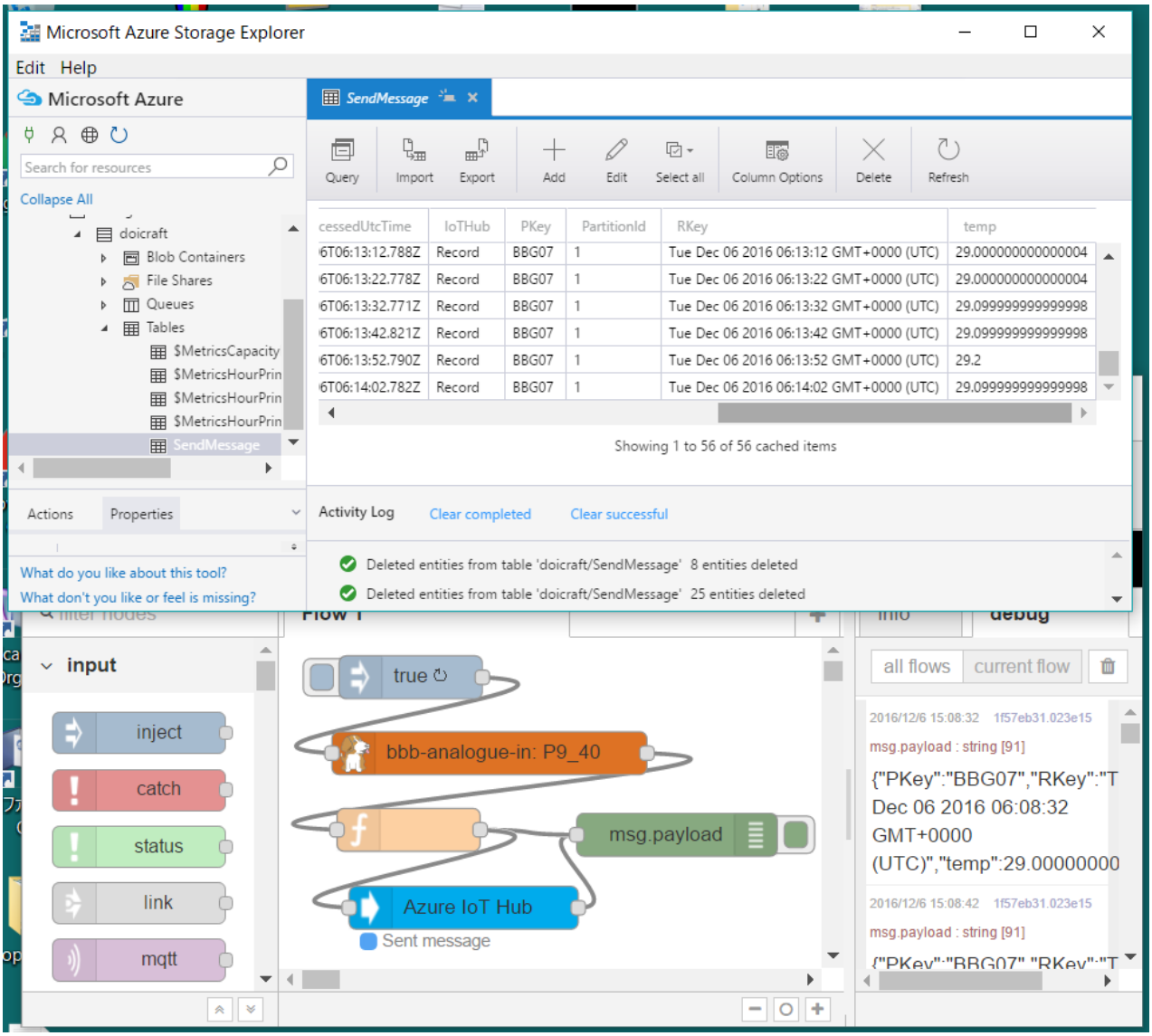

図 5.20　ストレージの状態と温度送信デバイスのフロー

～フロー8　/BBG

・コネクションストリングは DeviceExplorer の Management タグから得る。～

リスト 5.5　パーティションキーと行キーを追加するスクリプト

```
var date_obj = new Date();
var x = msg.payload;
var obj = '{"PKey":"BBG07","RKey":null,"temp":null}';
var obj2 = JSON.parse(obj);
obj2.RKey = date_obj.toString();
obj2.temp = x;
msg.payload =  JSON.stringify(obj2);
return msg;
```

(3) PowerBI でグラフ表示

Azure と連携する表示ツールに PowerBI があります。平たく言うと Excel のグラフ機能を拡張したようなツールと考えるとわかりやすいでしょう。

○PowerBI のインストール

PowerBI のサイトにいくとダウンロードできます。32bit 版と 64bit 版があり、使う PC に合わせたバージョンをインストールします。現状日本語のページからは「高度なダウンロード オプション」を選択しないと 32bit 版のリンクには着けません。

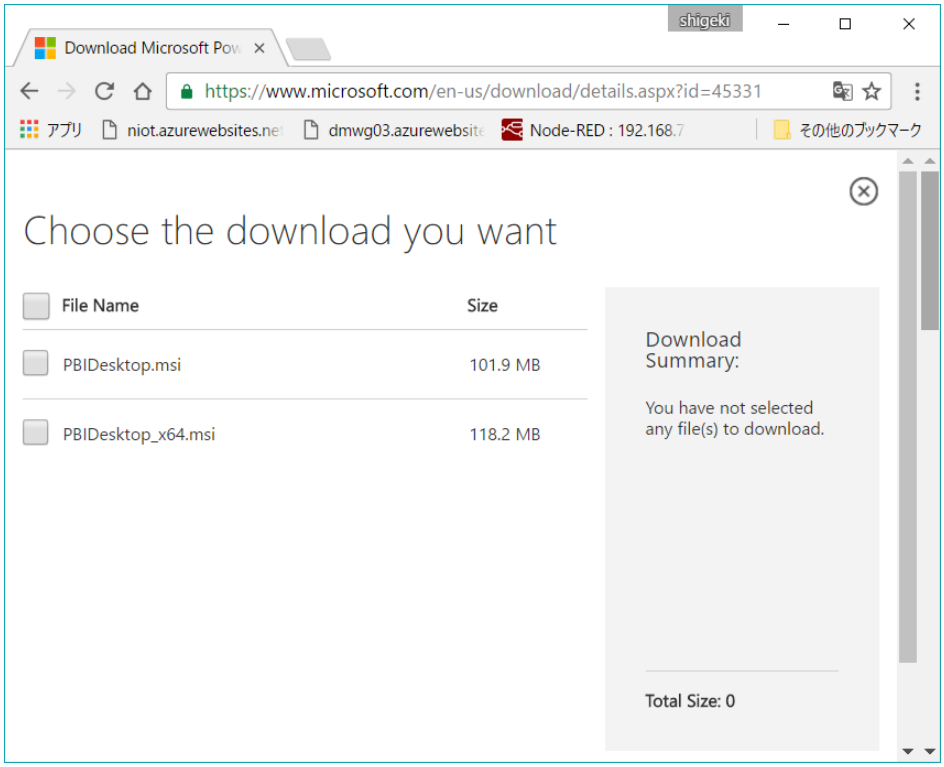

図 5.21　PowerBI のダウンロードページ

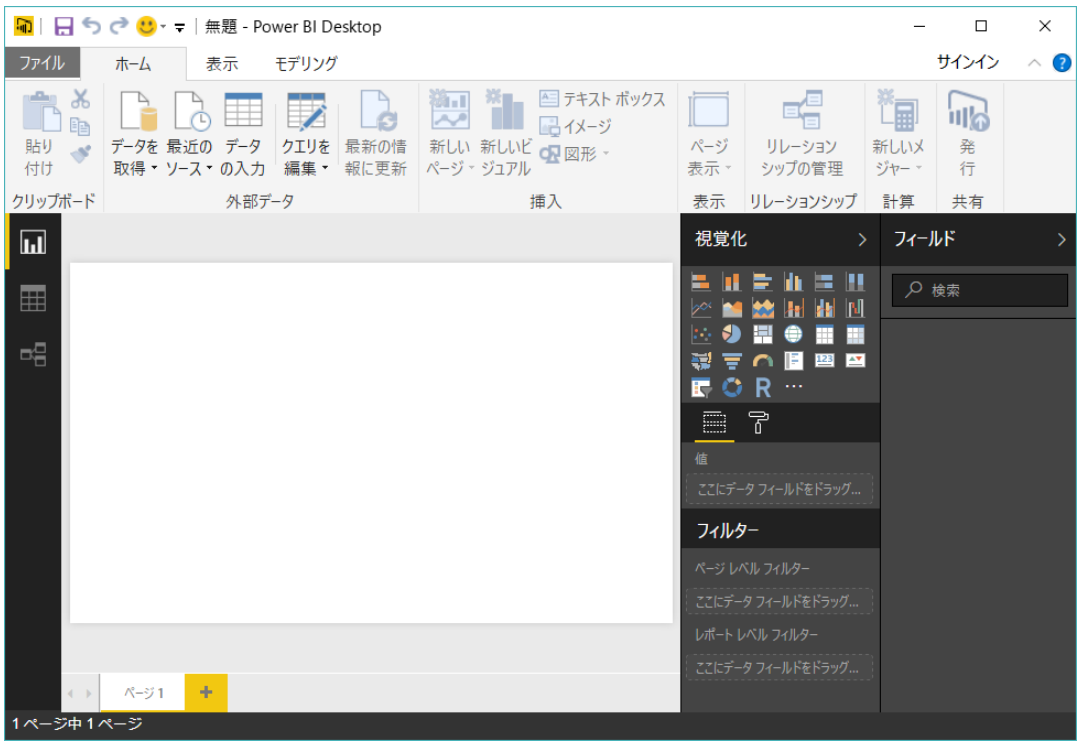

図 5.22　PowerBI

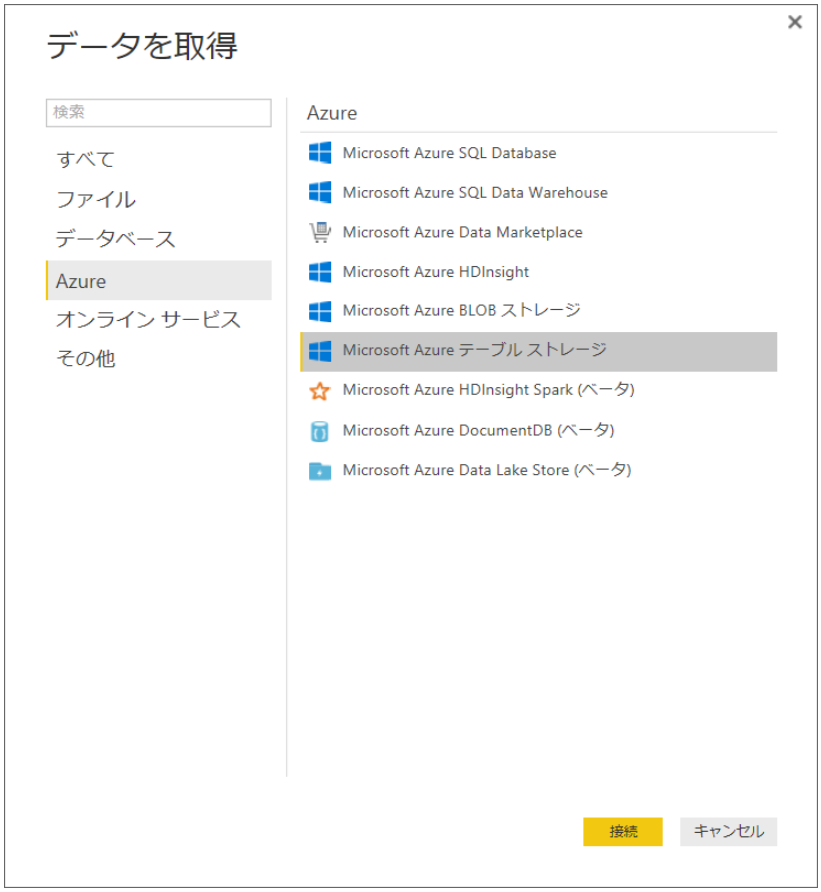

図 5.23　「データを取得」ウィンドウ

図 5.24　ストレージアカウント名を入力します。

図 5.25　ストレージアカウントキーを入力します。

〇PowerBI のクリエウィンドウの操作

「編集」からクリエウィンドウを開きます。

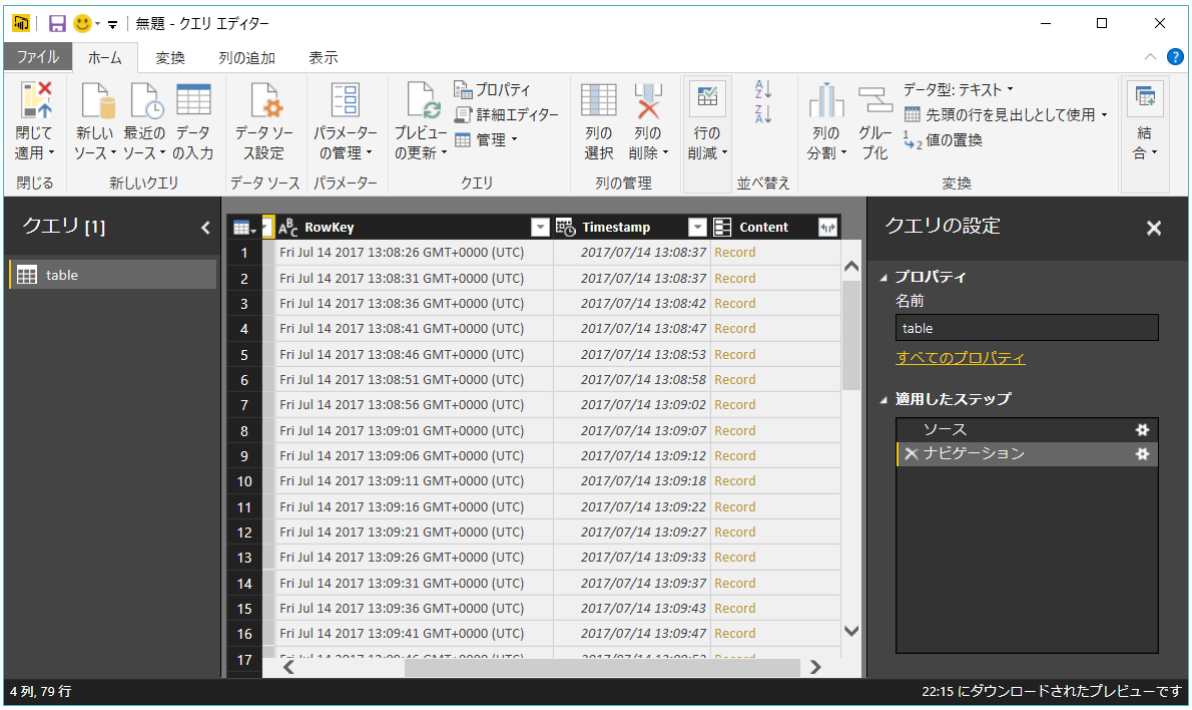

図 5. 26

文字列で得られている温度データを数値に変換します。

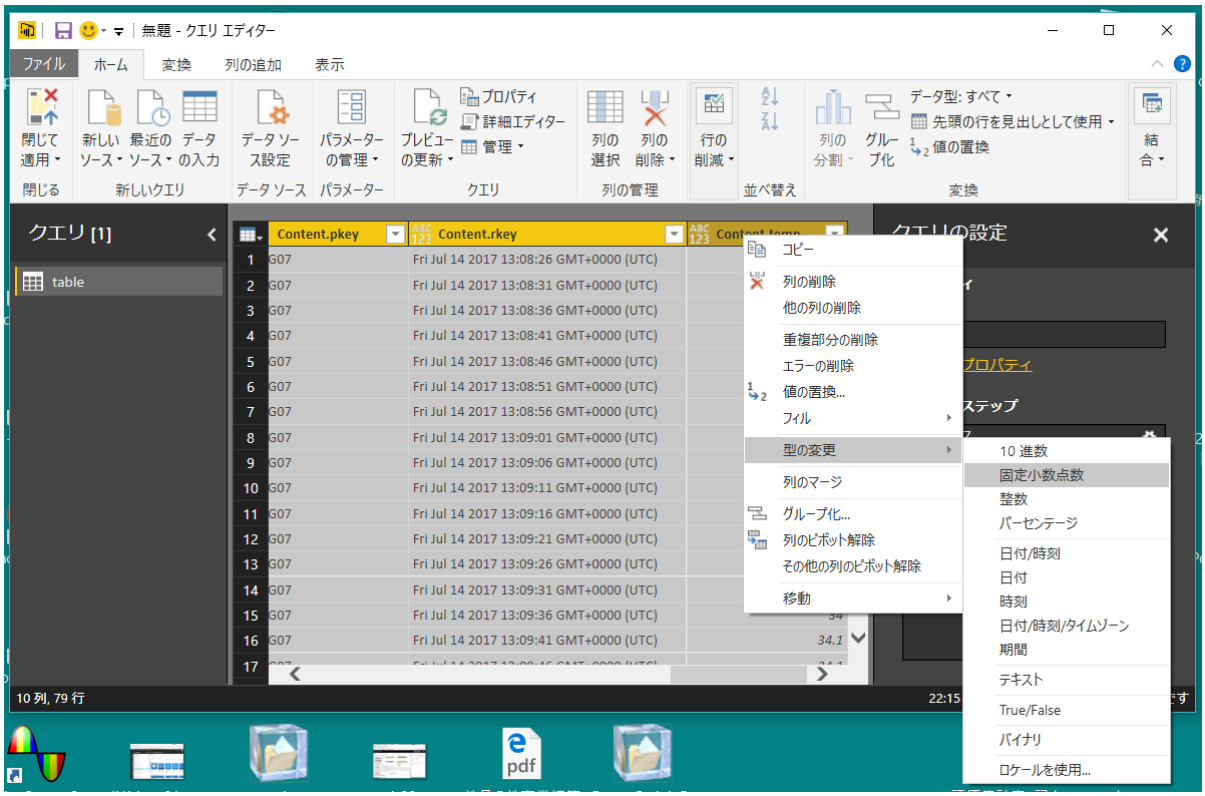

図 5. 27

「閉じて適応」をクリックして取り込みます。

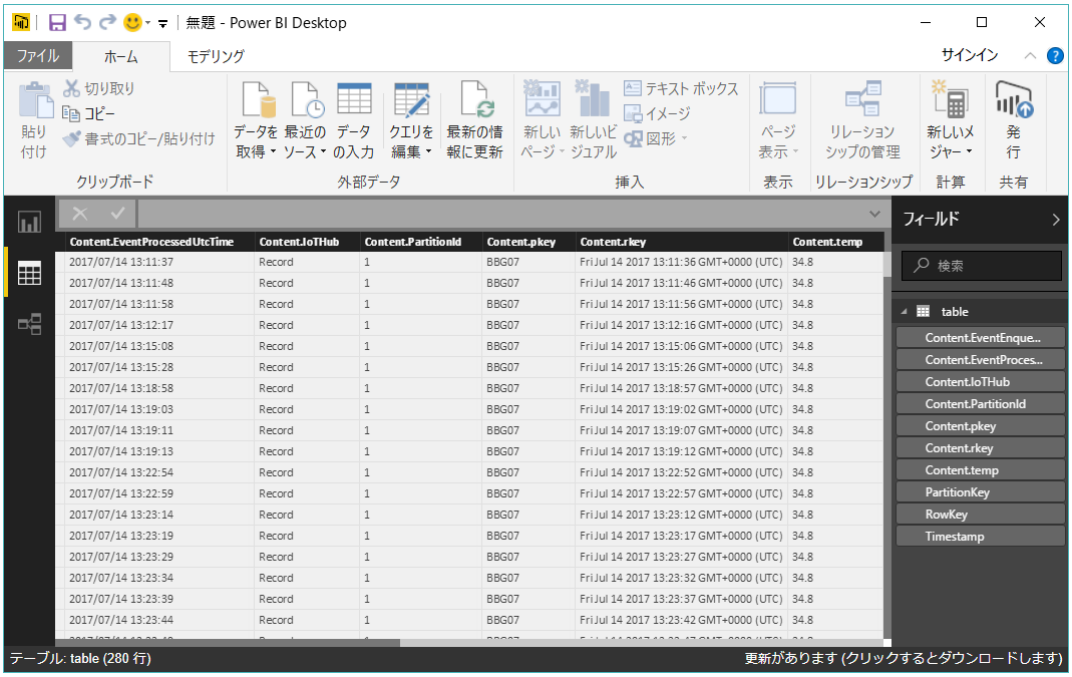

図 5. 28

グラフにしてみます。

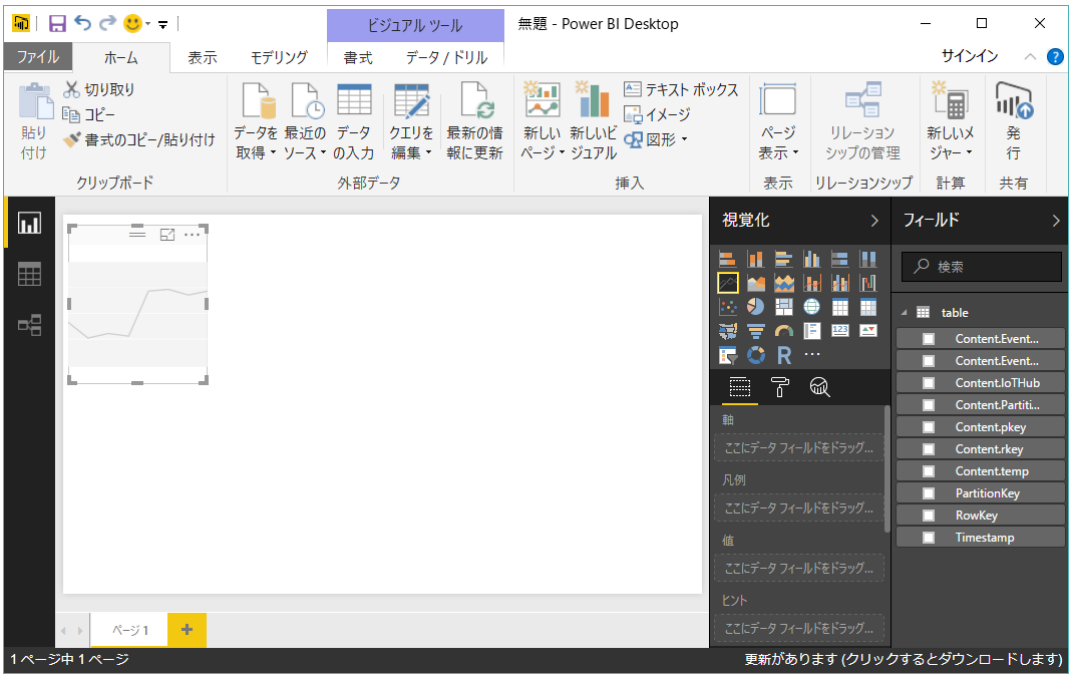

図 5. 29

データを「軸」と「値」にドラッグします。

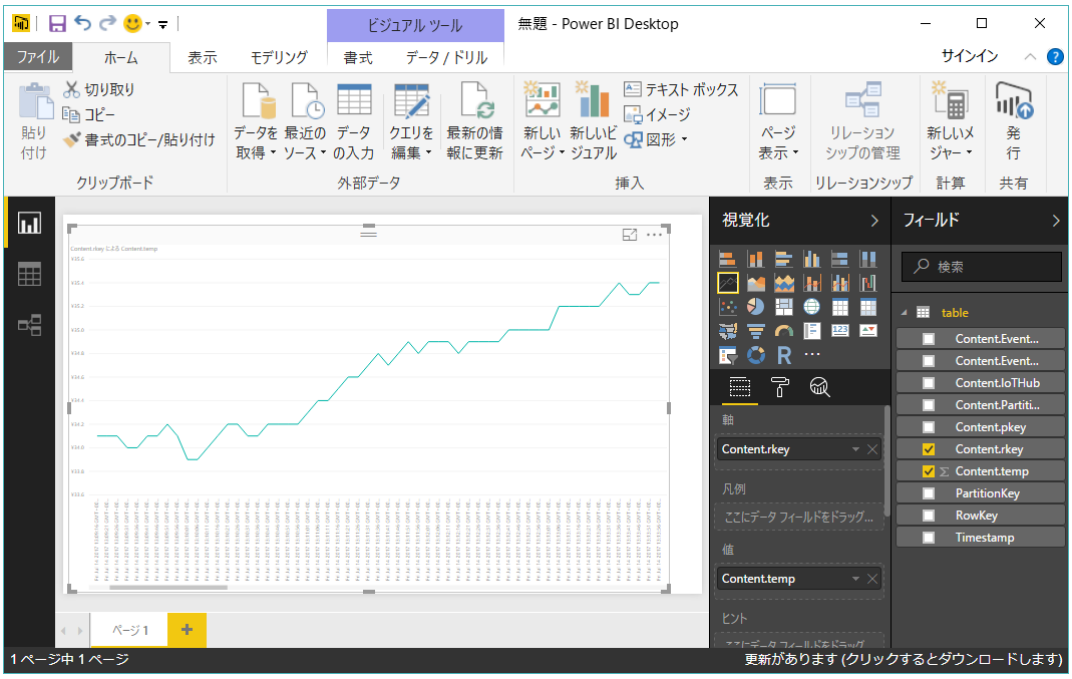

図 5.30

(4) アプリケーションからの読み出し

最後にアプリケーションからの読み出しを試します。アプリケーションから API を使ってストレージにアクセスすることができます。JavaScript/Node. js では azure-storage というライブラリを使えば TableStorage からデータを取り出せます。TableStorage に接続してクエリを作って読み出すという手順です。API には取り出しだけでなく様々な操作用意されています。

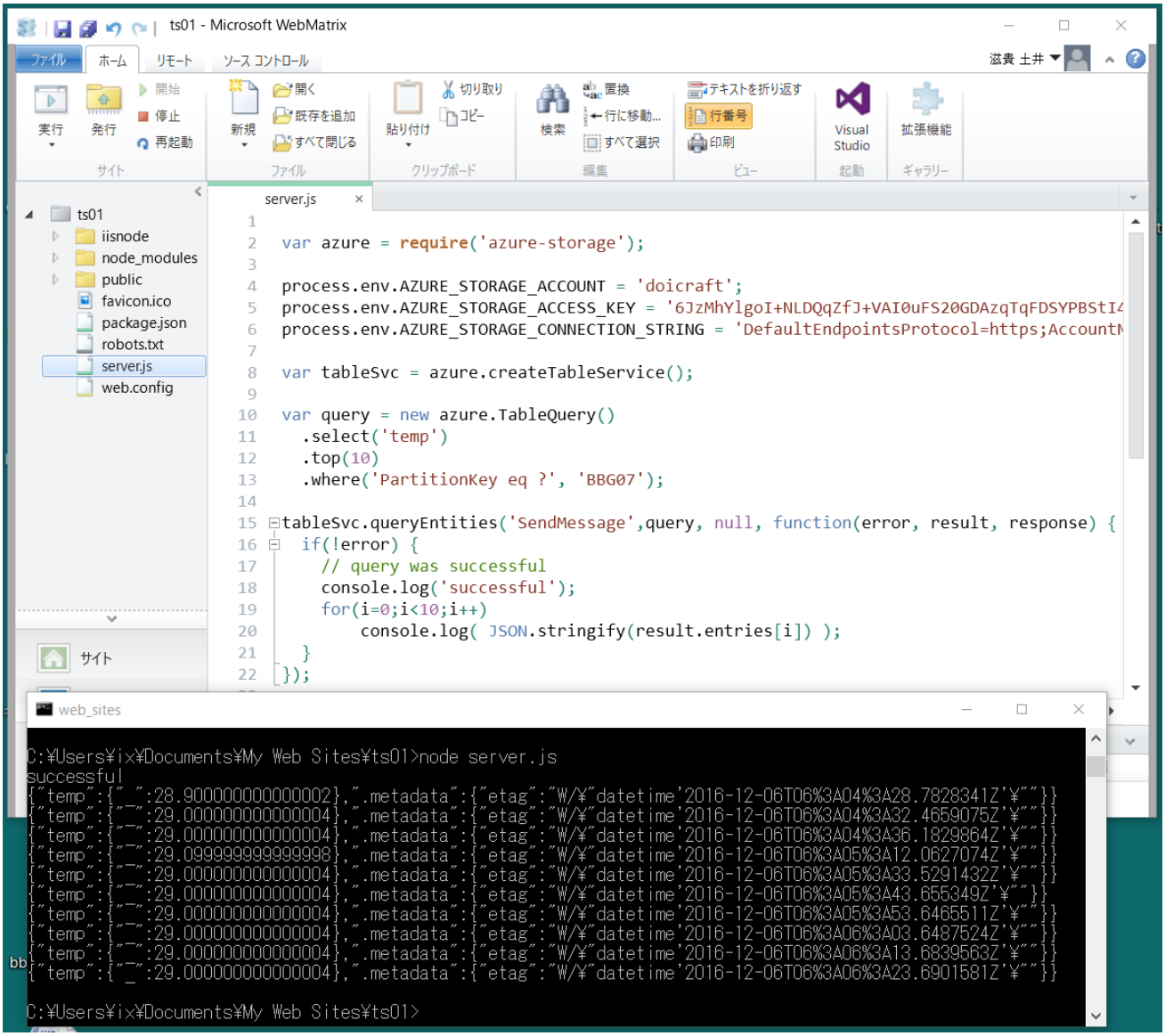

図 5.27 リスト 503 と実行の様子

あらかじめ、npm を使って azure-storage をインストールしておきます。

・ストレージ アカウント　は　作成したストレージアカウントの名前

・ストレージ アクセス キー　は「Key1」

・ストレージ コネクション ストリング　は

DefaultEndpointsProtocol=https;

AccountName=ストレージ名;

AccountKey=ストレージアクセスキー

を 1 行に生成します。

・tableSvc.queryEntities(xx ,,, の xx は作成したテーブルの名前

・WebMatrix を利用する場合、出力はコンソールを利用するので実行はコマンドプロンプトから行います。

リスト

```
var azure = require('azure-storage');

process.env.AZURE_STORAGE_ACCOUNT = 'dlts00';
process.env.AZURE_STORAGE_ACCESS_KEY = 'KQqCMzop52XP4LqL1nBXyDDd/xxxx';
process.env.AZURE_STORAGE_CONNECTION_STRING =
'DefaultEndpointsProtocol=https;AccountName=dlts00;AccountKey=KQqCMzop52XP4LqL1nBXyDDd/xxxx';

var tableSvc = azure.createTableService();

var query = new azure.TableQuery()
  .select('RKey') // temp')
  .top(10)
  .where('PartitionKey eq ?', 'PC');//BBG07');

tableSvc.queryEntities('iotdata',//'SendMessage',
           query, null, function(error, result, response) {
  if(!error) {
    // query was successful
    console.log('successful');
    for(i=0;i<10;i++)
        console.log( JSON.stringify(result.entries[i]) );
  }
});
```

(5)Excel からの呼び出し

Excel の 2010 以降ですと、アドオンを追加、あるいは標準でテーブルストレージ他の Azure のストレージにアクセスすることができます。ここでは 2013 を例に紹介します。

○アドオンのインストール

マイクロソフトのサイトからアドオンをダウンロードします。

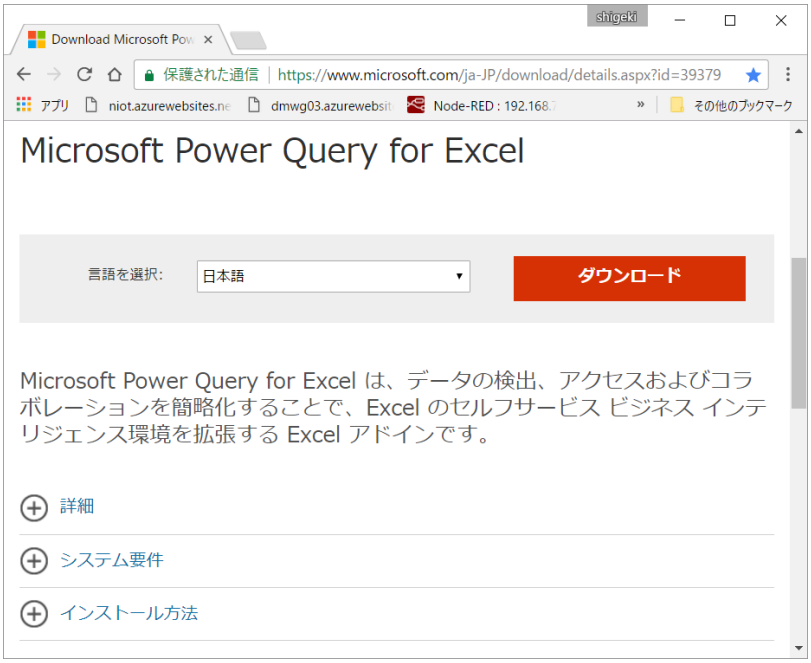

図 5.28

インストールが済めば、Azure のストレージからの読み出しメニューが追加されます。

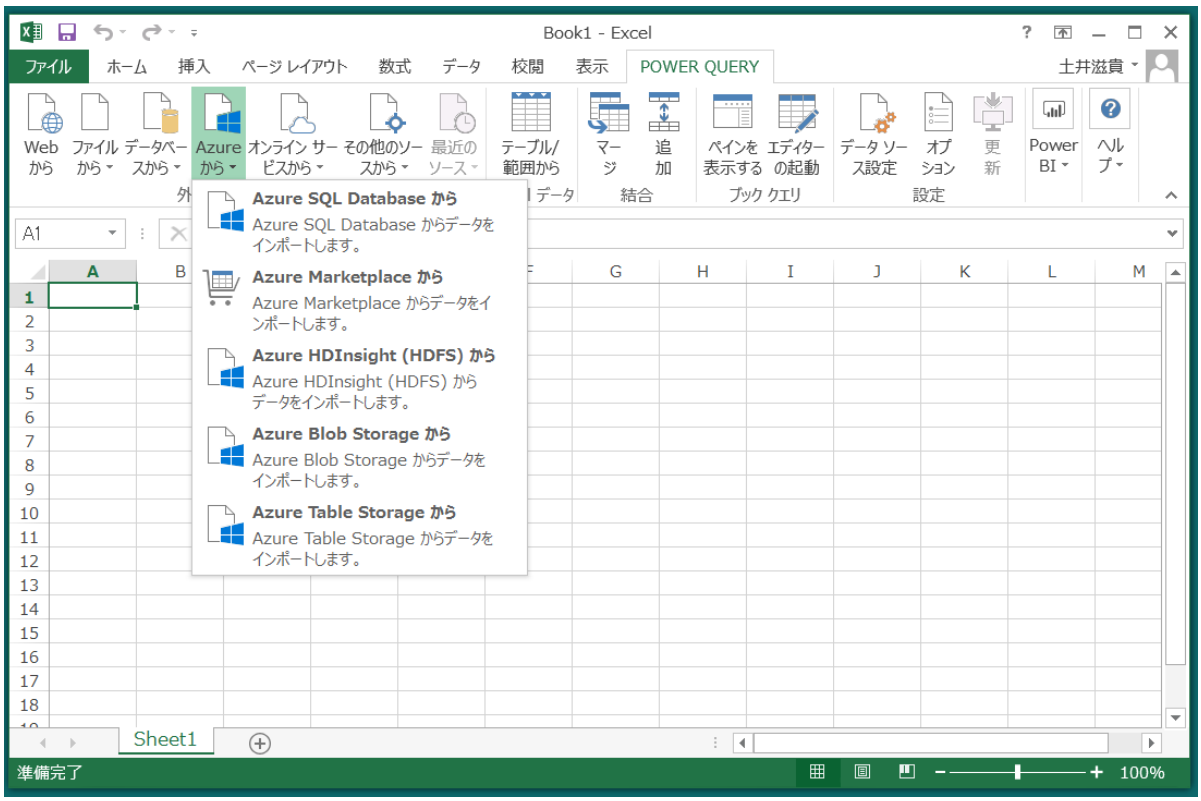

図 5.29

そのうちから、「TableStorage からインポート」選択します。するとストレージアカウントの入力ウィンドウが開きます。ここで、ストレージアカウント名を入力します。その後、更にキーを聞いてきますので、入力します。

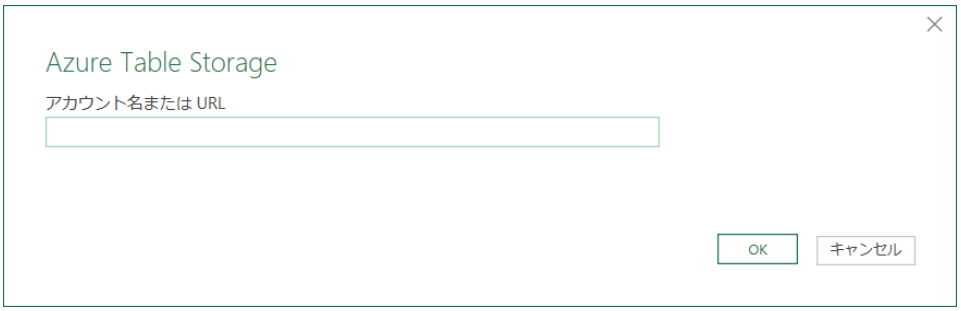

図 5.30

そうすると、指定したストレージアカウントの TableStorage が表示されます。データ本体は「Recored」にまとまって示されます。これを展開するために「編集」をクリックします。

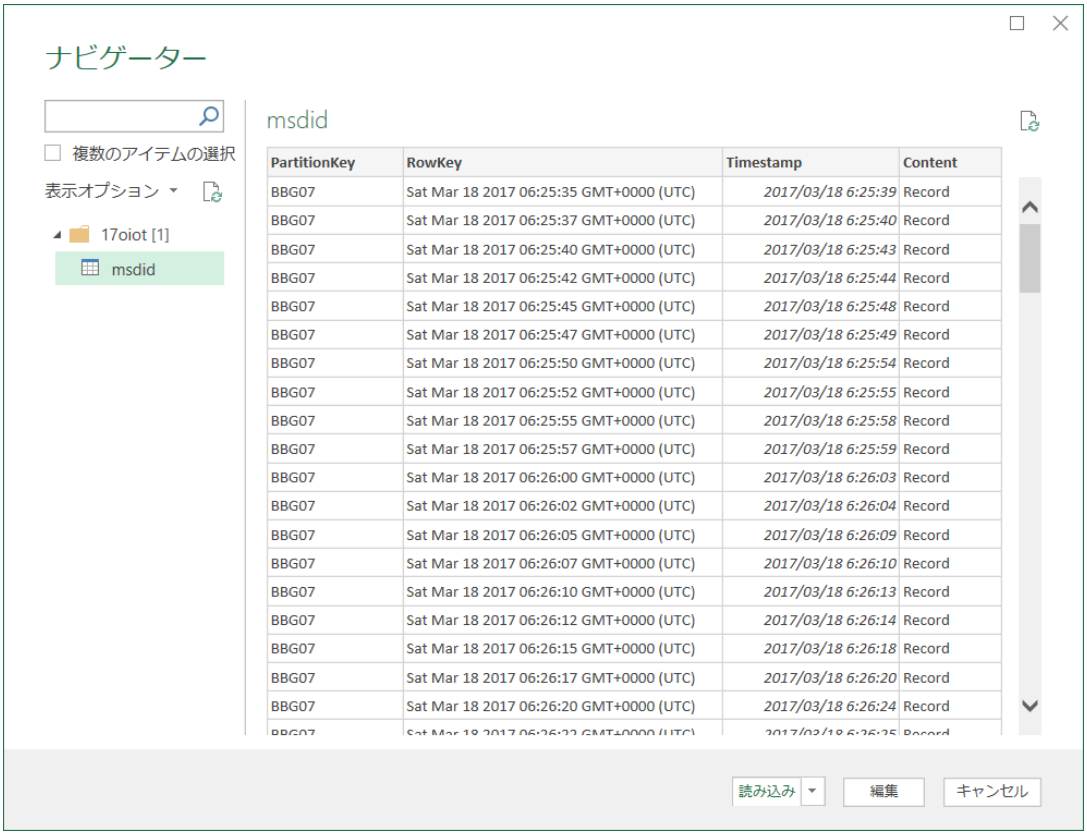

図 5.31

クリエ エディタと呼ばれるウィンドウが開きます。とりあえず、すべて展開で「OK」をクリックします。

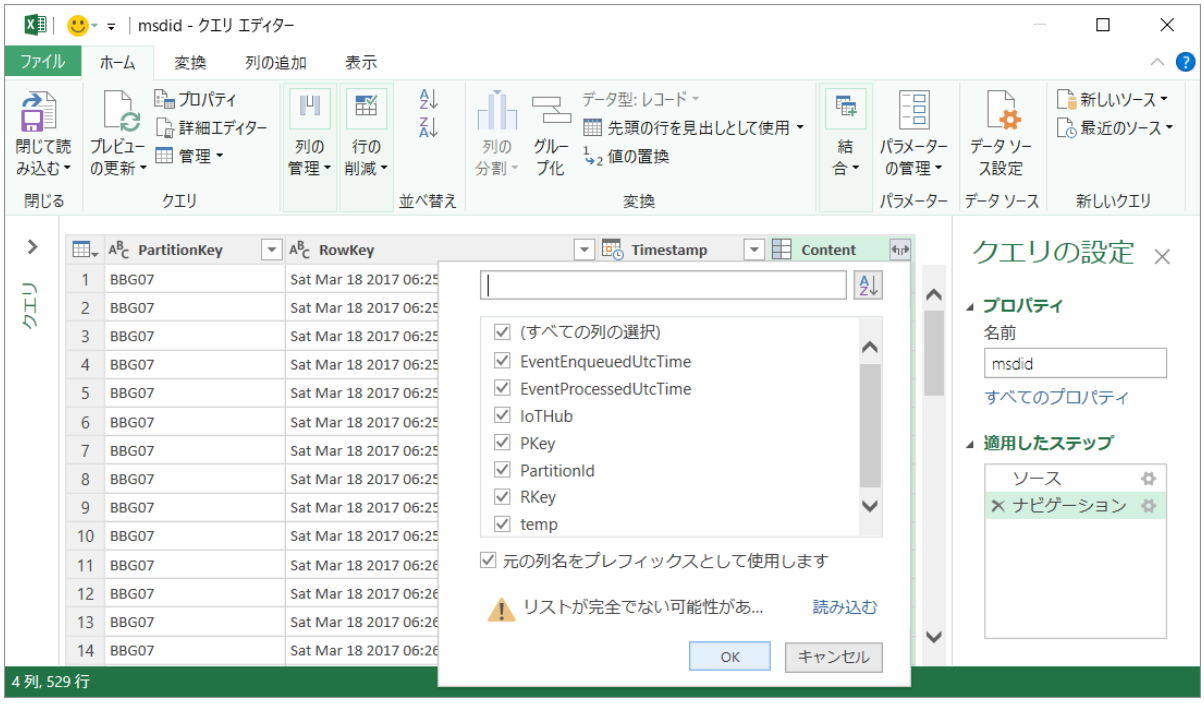

図 5.32

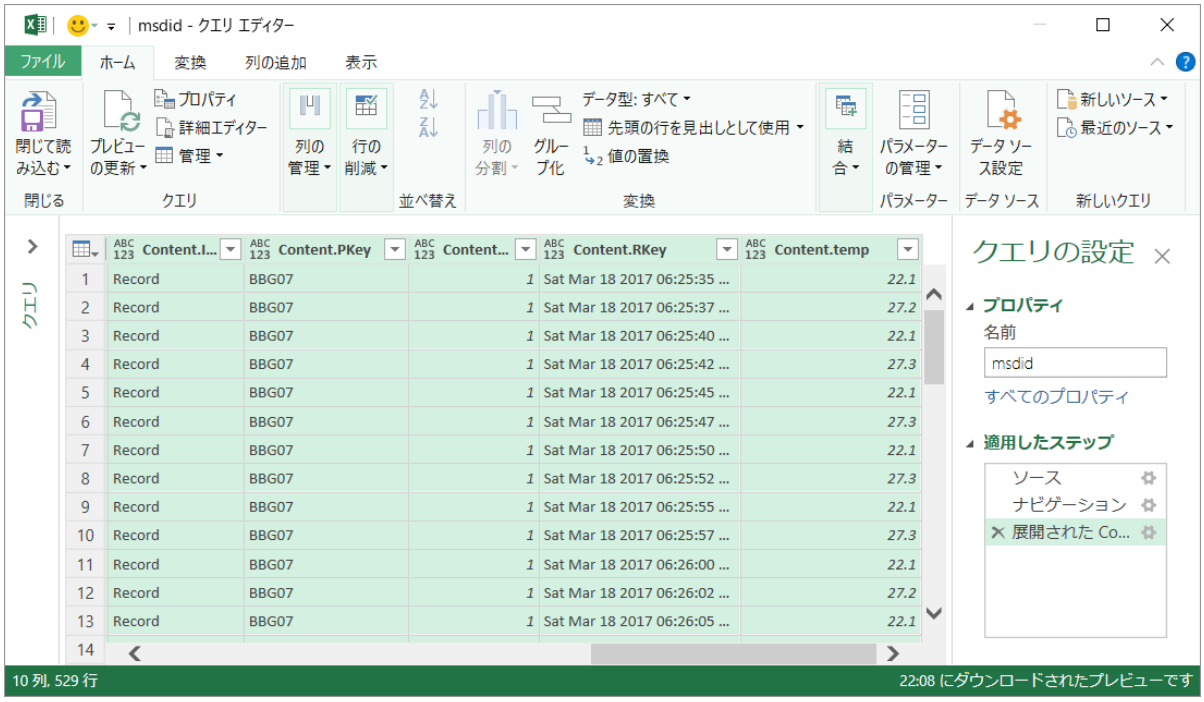

図 5. 33

データが展開されました。「閉じて読み込む」クリックします。クリエウィンドウの内容がシートに展開されました。

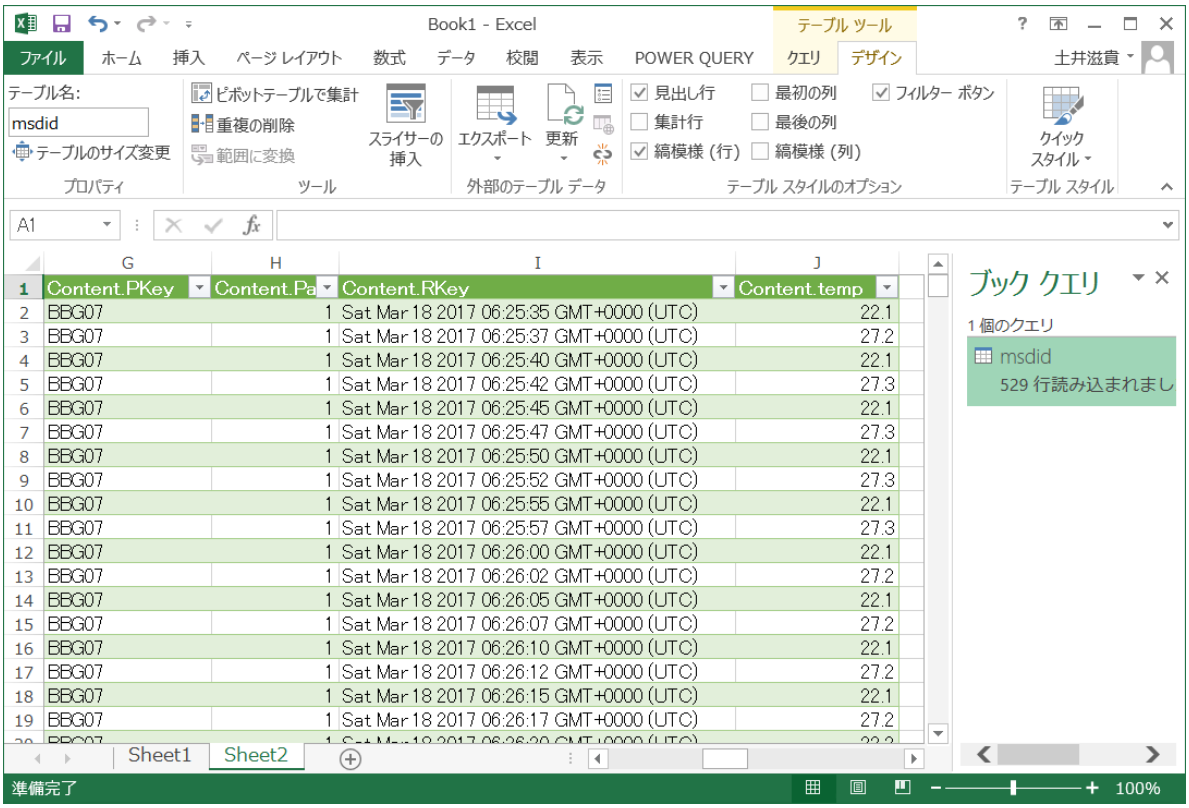

図 5. 34

5.3 Azure Cognitive Services を使う

最後に各種認識などクラウドに用意されているより上位な機能を試します。今回はこのなかから、音声-テキスト変換の Bing Speech API と Computer Vision API を試します。

(1) Azure Cognitive Services

各クラウドともこのようなサービスが準備されています。Azure の場合は、CognitiveServices と呼ばれるグループで扱われます。

図 5.28　Cognitive Services のページ

API については図 5.26　の　cognitive services　ページ

https://www.microsoft.com/cognitive-services

があります。

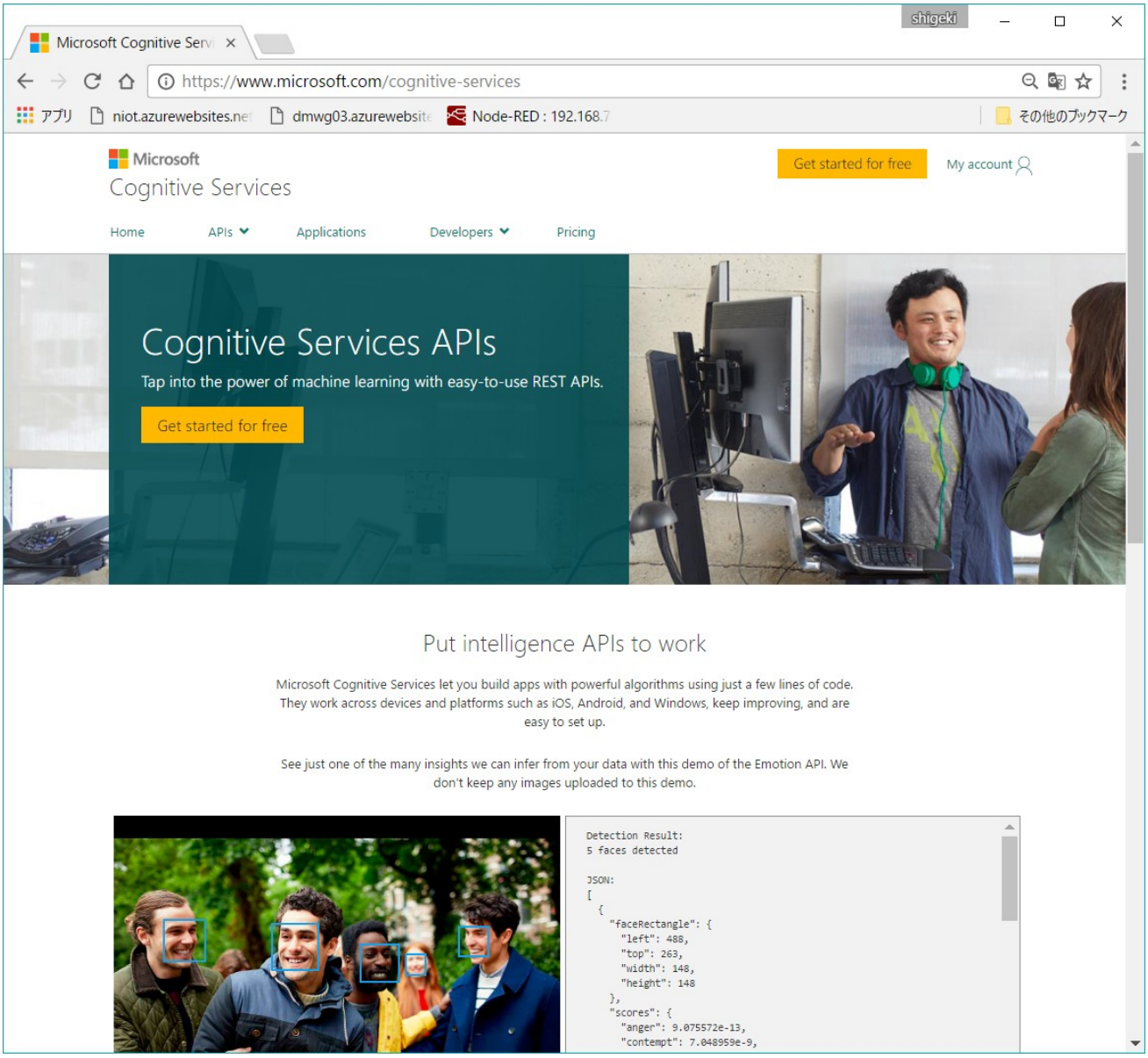

図 5.29　cognitive services　ページ　https://www.microsoft.com/cognitive-services

このページから Developers の SDKs & Samples に移動すると、図 5.27　の　SDKs & Samples　ページ

https://www.microsoft.com/cognitive-services/en-us/SDK-Sample

があります。

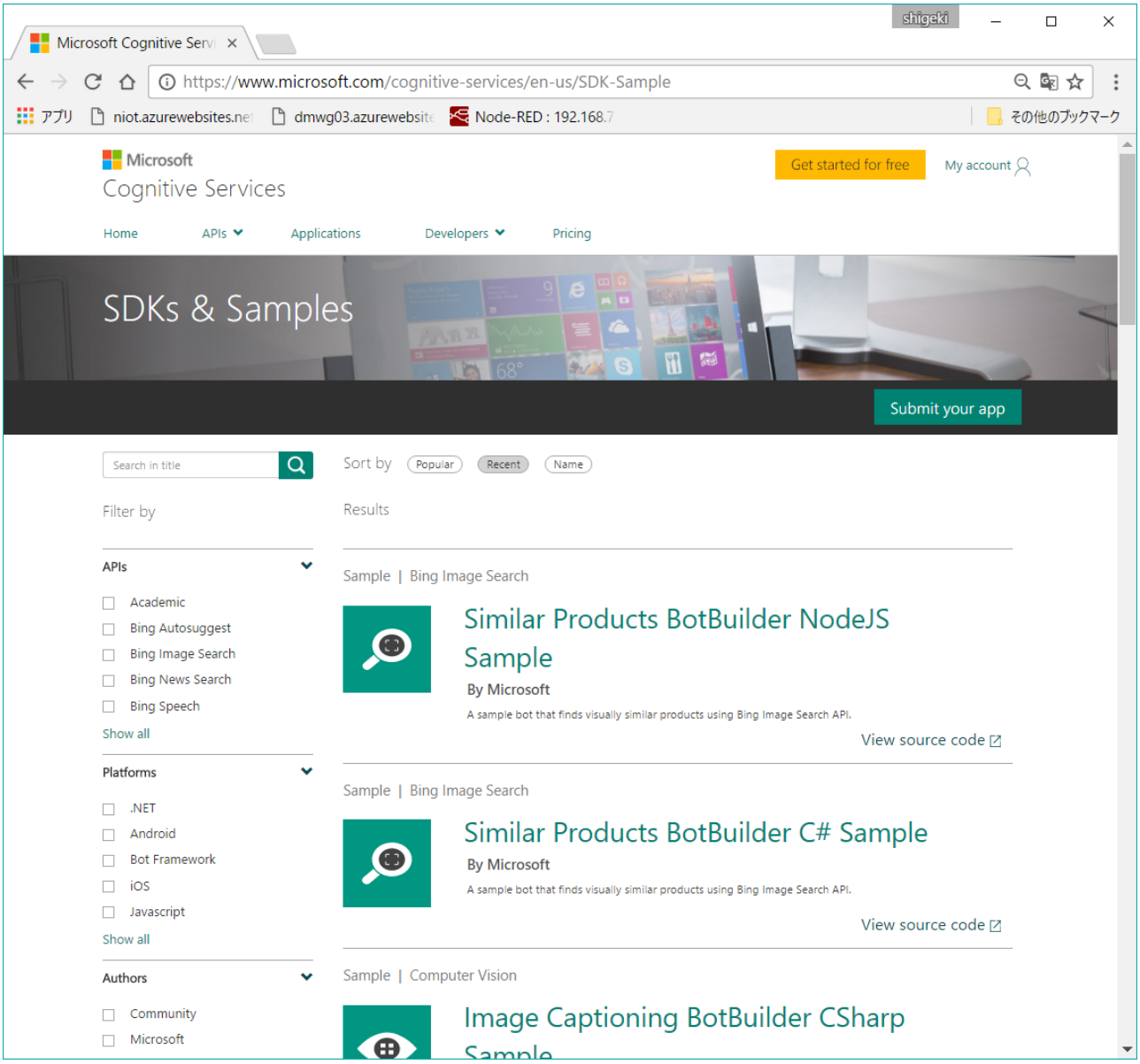

図 5.30　SDKs & Samples　ページ　https://www.microsoft.com/cognitive-services/en-us/SDK-Sample

(2) Bing Speech を試す

このページにの Bing Speech-to-Text Javascript SDK に行くと、図 5.28 の github の Microsoft/Cognitive-Speech-STT-JavaScript ページがあります。

https://github.com/Microsoft/Cognitive-Speech-STT-JavaScript

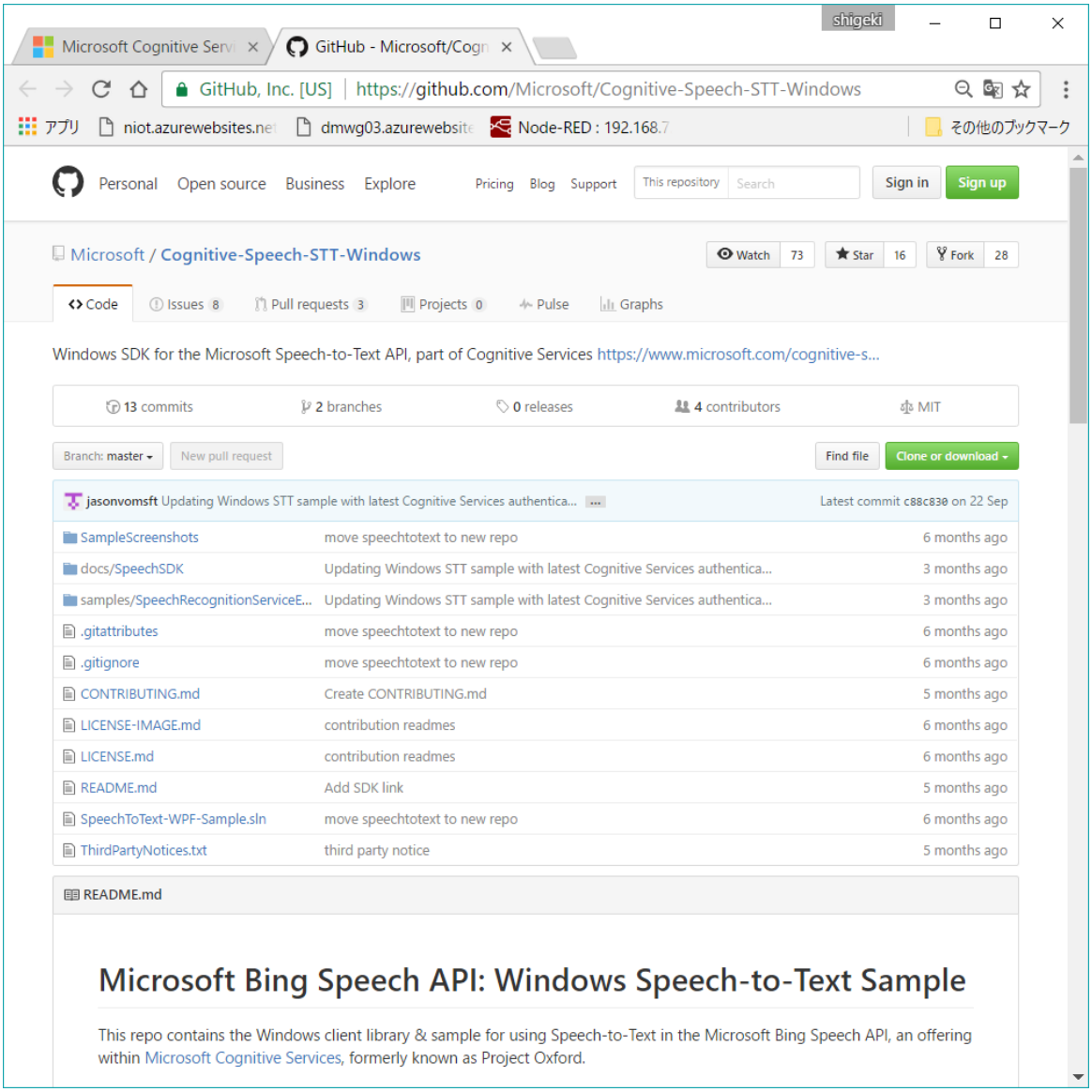

図 5.31　Microsoft/Cognitive-Speech-STT-JavaScript　ページ。

https://github.com/Microsoft/Cognitive-Speech-STT-JavaScript

(2) Cogniteve Services の登録

「AI + Cognitive Service から「See all」をクリックし、「Bing Speech API」を選択します。

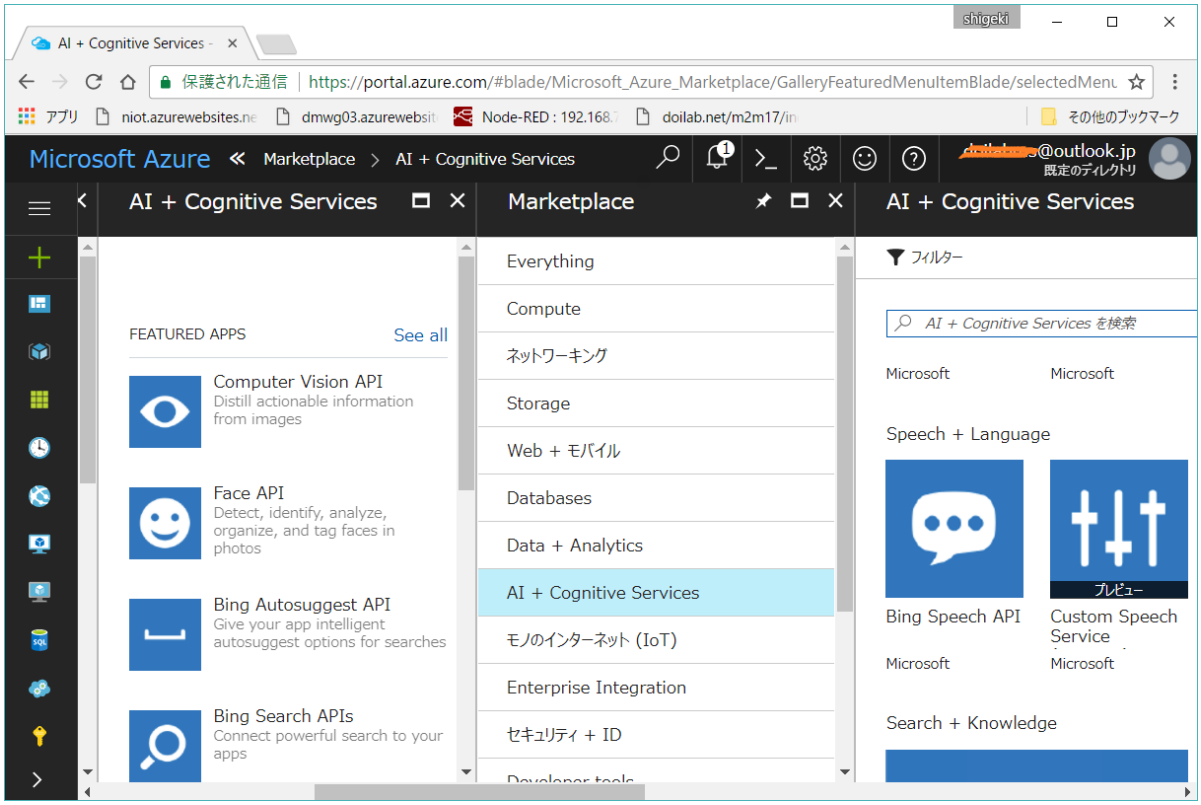

図 5.

名前、価格レベル、リソースグループを設定して作成します。

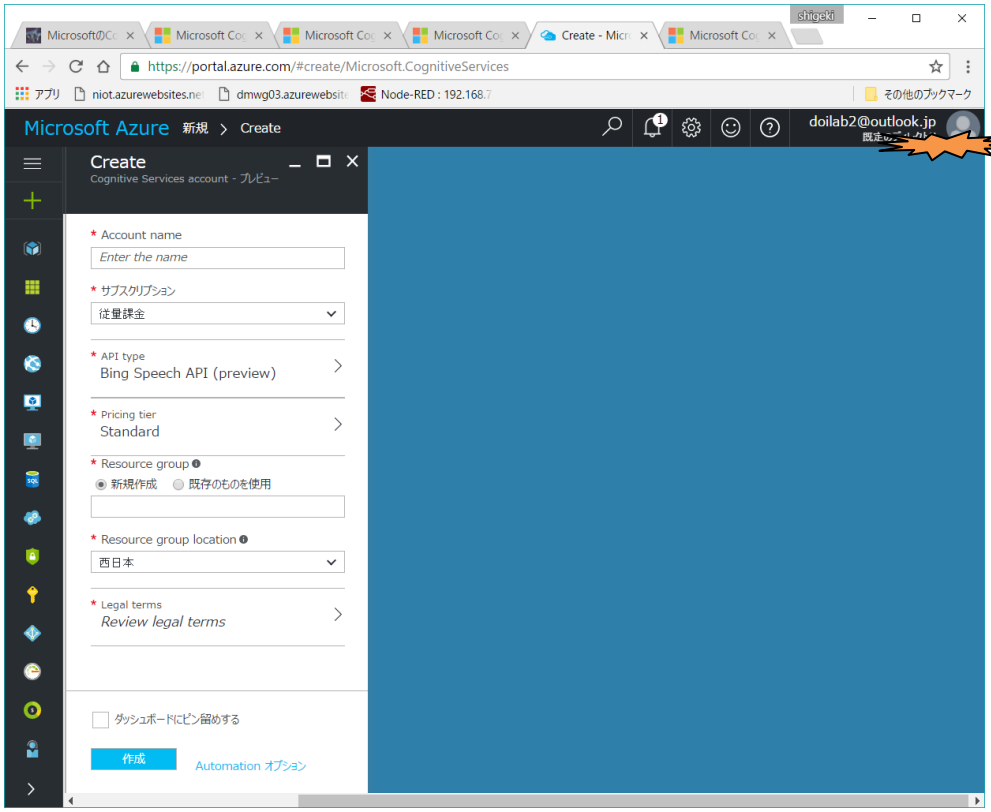
https://portal.azure.com/#create/Microsoft.CognitiveServices
Microsoft Azure 新規 > Create
doilab2@outlook.jp
Create
Cognitive Services account - プレビュー
* Account name
Enter the name
* サブスクリプション
従量課金
* API type
Bing Speech API (preview)
* Pricing tier
Standard
* Resource group
新規作成
既存のものを使用
* Resource group location
西日本
* Legal terms
Review legal terms
ダッシュボードにピン留めする
作成
Automation オプション

図 5. 35

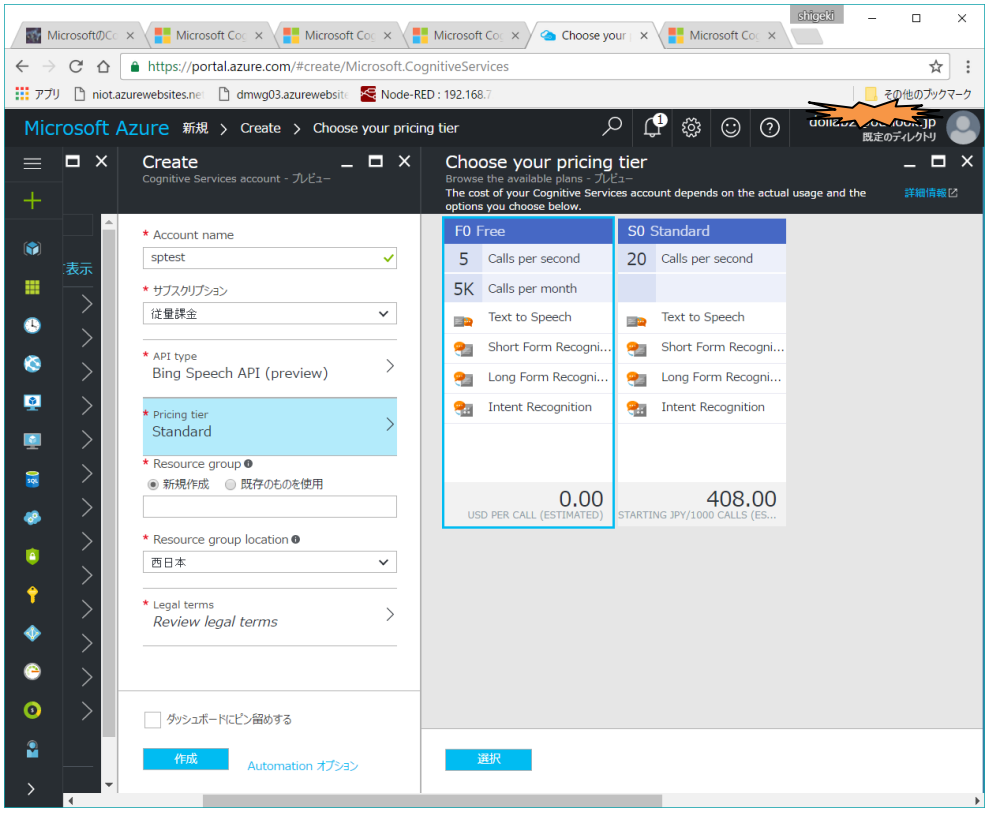
https://portal.azure.com/#create/Microsoft.CognitiveServices
Microsoft Azure 新規 > Create > Choose your pricing tier
Create
Cognitive Services account - プレビュー
Choose your pricing tier
Browse the available plans - プレビュー
The cost of your Cognitive Services account depends on the actual usage and the options you choose below.
詳細情報
* Account name
sptest
* サブスクリプション
従量課金
* API type
Bing Speech API (preview)
* Pricing tier
Standard
* Resource group
新規作成
既存のものを使用
* Resource group location
西日本
* Legal terms
Review legal terms
ダッシュボードにピン留めする
作成
Automation オプション
F0 Free
5 Calls per second
5K Calls per month
Text to Speech
Short Form Recogni...
Long Form Recogni...
Intent Recognition
0.00
USD PER CALL (ESTIMATED)
S0 Standard
20 Calls per second
Text to Speech
Short Form Recogni...
Long Form Recogni...
Intent Recognition
408.00
STARTING JPY/1000 CALLS (ES...
選択

図 5. 36

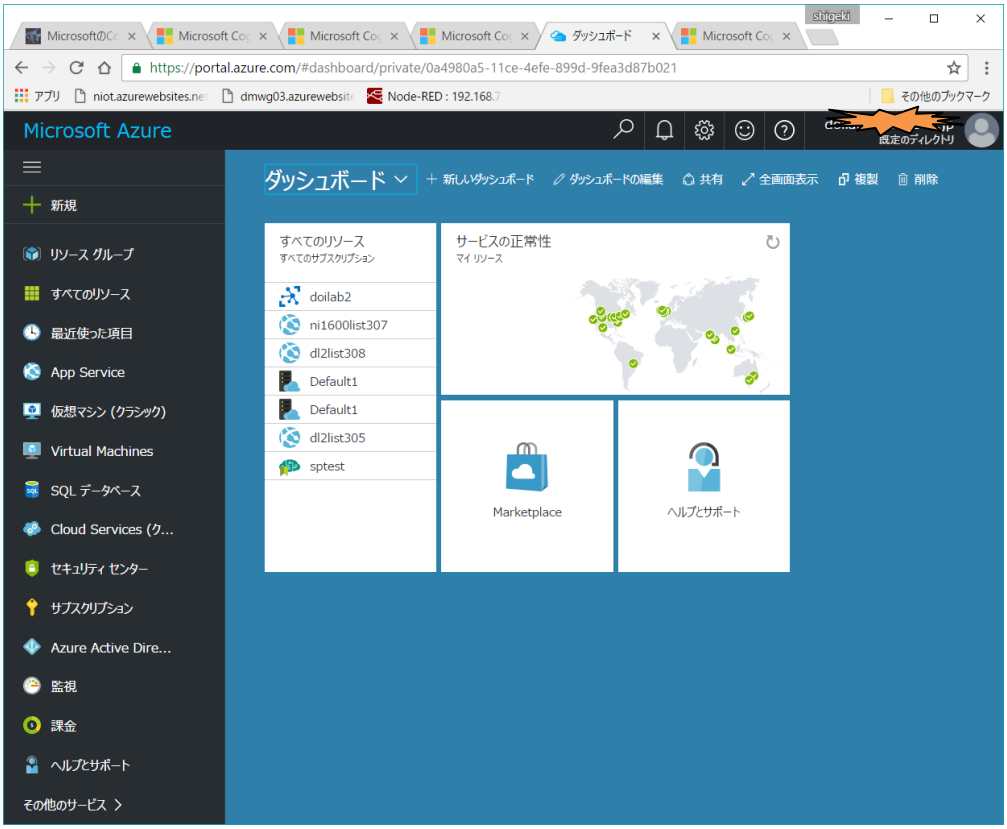

図 5.37

これで図 5.34 に示すようにサービスが生成されます。サンプルのアプリを動かすために図 5.35、図 5.36 に示すメニューからキーを取得します。

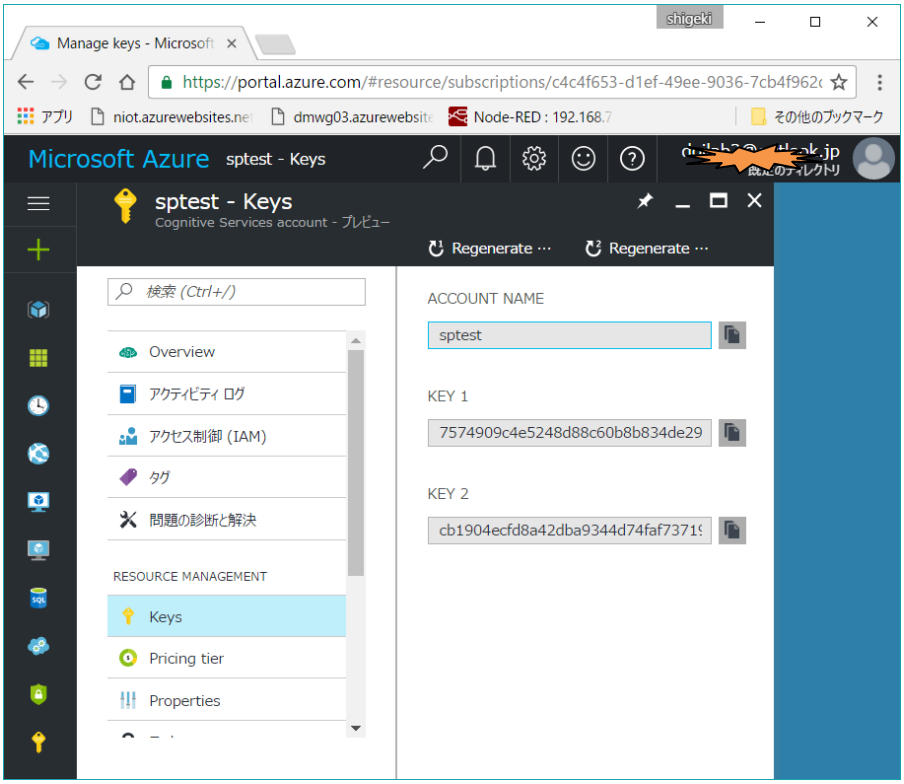
shigeki
Manage keys - Microsoft
https://portal.azure.com/#resource/subscriptions/c4c4f653-d1ef-49ee-9036-7cb4f962
アプリ
niot.azurewebsites.net
dmwg03.azurewebsite
Node-RED : 192.168.7
その他のブックマーク
Microsoft Azure
sptest - Keys
既定のディレクトリ
sptest - Keys
Cognitive Services account - プレビュー
Regenerate ···
Regenerate ···
検索 (Ctrl+/)
Overview
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
RESOURCE MANAGEMENT
Keys
Pricing tier
Properties
ACCOUNT NAME
sptest
KEY 1
7574909c4e5248d88c60b8b834de29
KEY 2
cb1904ecfd8a42dba9344d74faf7371

図 5. 38

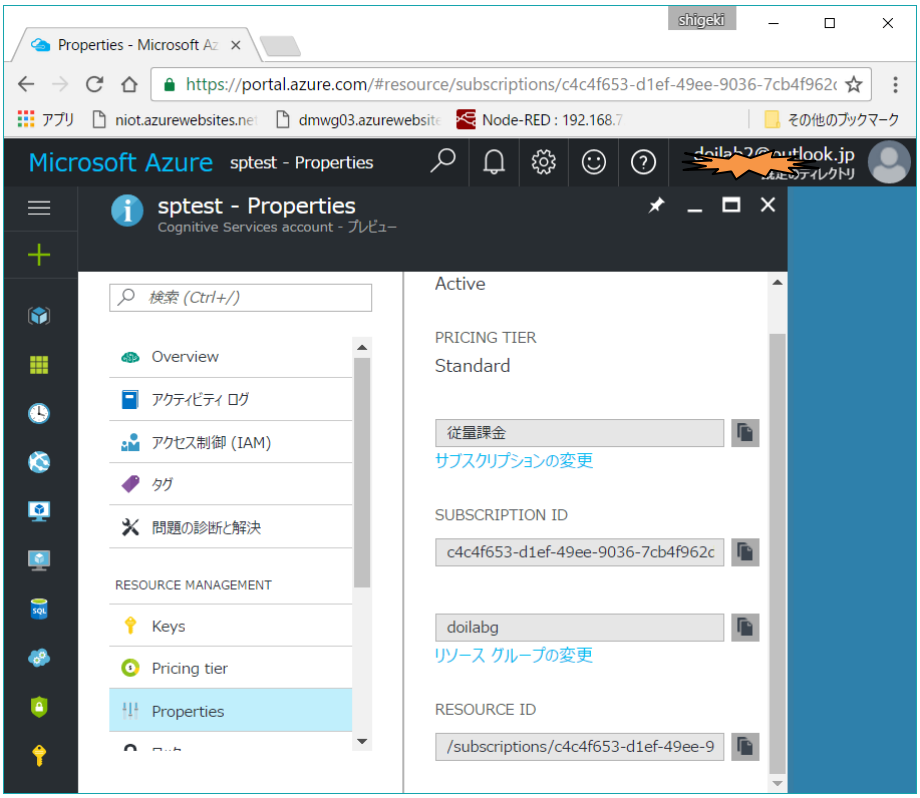
shigeki
Properties - Microsoft Az
https://portal.azure.com/#resource/subscriptions/c4c4f653-d1ef-49ee-9036-7cb4f962
アプリ
niot.azurewebsites.net
dmwg03.azurewebsite
Node-RED : 192.168.7
その他のブックマーク
Microsoft Azure
sptest - Properties
既定のディレクトリ
sptest - Properties
Cognitive Services account - プレビュー
検索 (Ctrl+/)
Overview
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
RESOURCE MANAGEMENT
Keys
Pricing tier
Properties
Active
PRICING TIER
Standard
従量課金
サブスクリプションの変更
SUBSCRIPTION ID
c4c4f653-d1ef-49ee-9036-7cb4f962c
doilabg
リソース グループの変更
RESOURCE ID
/subscriptions/c4c4f653-d1ef-49ee-9

図 5. 39

(3) シンプルなプログラムを試す。

本プログラムは Node. js のプロジェクトでなく通常の HTML プロジェクトとして作成します。

WebMatrix で「HTML」の「空のサイト」のテンプレートを選択しアプリを作ります。

また、IE11 や Chrome ではマイク入力がうまく動きません。FireFox を試してみてください。

https://www.mozilla.jp/

リスト 7

・managekey　は　「KEY1」

・リソース ID　は　「Propertise」の「RESOURCE ID」

・サブスクリプション ID　は　「Propertise」の「SUBSCRIPTION ID」

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <script src="./speech.1.0.0.js" type="text/javascript"></script>
    <script type="text/javascript">
        var client;
        var request;

        function clearText() {
            document.getElementById("output").value = "";
        }
        function setText(text) {
            document.getElementById("output").value += text;
        }
        function start() {
            client =
Microsoft.CognitiveServices.SpeechRecognition.SpeechRecognitionServiceFactory.createMicrophoneClientWithIntent(
                'en-us',                                  // Language
                '7574909c4e5248d88cxxxxx',     // manage key

'/subscriptions/cxxxx/resourceGroups/doilabg/providers/Microsoft.CognitiveServices/accounts/sptest',
                                                          // RESOURCE ID
                'c4c4f653-d1ef-49ee-xxxxxxxxx'  // SUBSCRIPTION ID
                          );
            client.startMicAndRecognition();
            setTimeout(function () {
                client.endMicAndRecognition();
            }, 5000);

            client.onPartialResponseReceived = function (response) {
                setText(response);
            }
            client.onFinalResponseReceived = function (response) {
                setText(JSON.stringify( response ));
```

```
            }
            client.onIntentReceived = function (response) {
                setText(response);
            }
        }
    </script>
</head>
<body style="font-family: Geneva,  sans-serif">
    <table style="width:100%">
        <tr>
            <td><button onclick="start()">Start</button></td>
        </tr>
        <tr>
            <td>
                <textarea id="output" style='width:400px;height:200px'></textarea>
            </td>
        </tr>
    </table>
</body>
</html>
```

サンプルプログラムに設定する各 ID・キー。

```
                function setText(text) {
                    document.getElementById("output").value += text;
                }
                function start() {
                    client =
        Microsoft.CognitiveServices.SpeechRecognition.SpeechRecognitionServiceFactory.createMicrophoneClientWithIntent(
                        'en-us',                                // Language
                        '7574909c4e5248d88cxxxxx',     // manage key
        '/subscriptions/cxxxx/resourceGroups/doilabg/providers/Microsoft.CognitiveServices/accounts/sptest',
                                                                // RESOURCE ID
                        'c4c4f653-d1ef-49ee-xxxxxxxxx'  // SUBSCRIPTION ID
                        );
                    client.startMicAndRecognition();
                    setTimeout(function () {
                        client.endMicAndRecognition();
                    }, 5000);
```

BingSpeechAPIのキー

サブスクリプションID

リソースグループ名

アカウント名

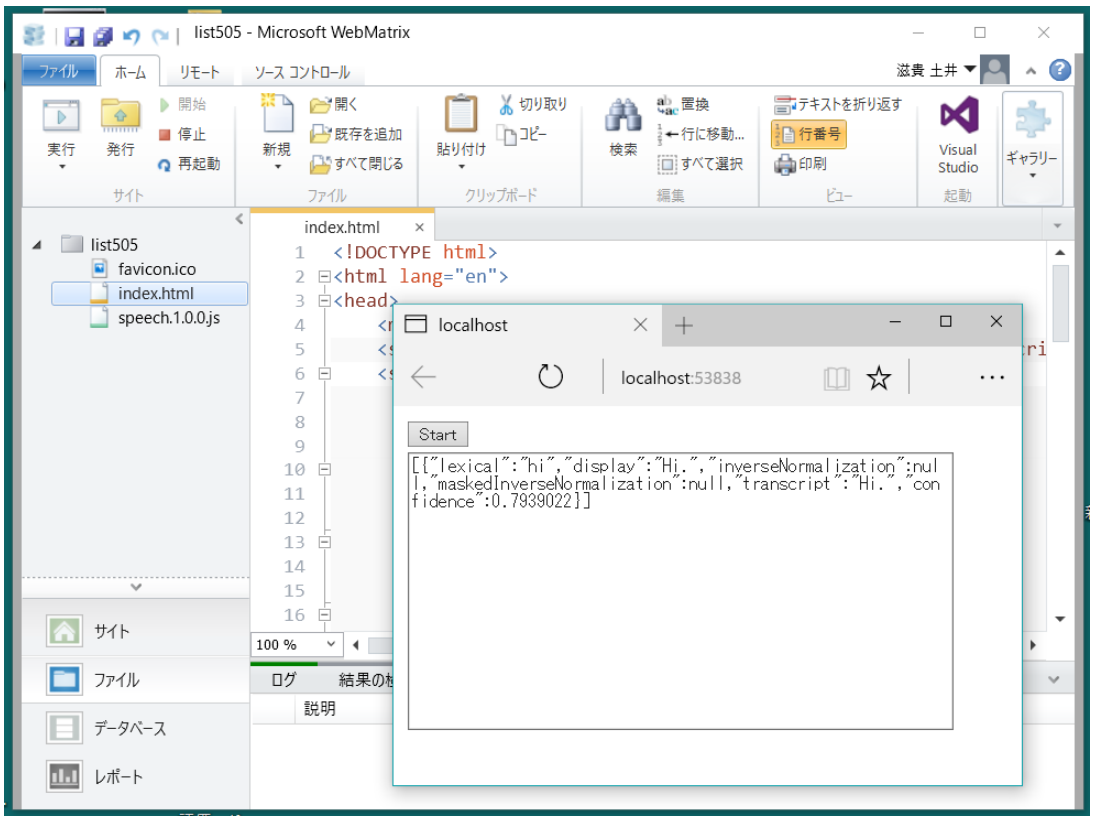
list505 - Microsoft WebMatrix
ファイル
ホーム
リモート
ソース コントロール
滋貴 土井
実行
発行
開始
停止
再起動
サイト
新規
開く
既存を追加
すべて閉じる
ファイル
貼り付け
切り取り
コピー
クリップボード
検索
置換
行に移動...
すべて選択
編集
テキストを折り返す
行番号
印刷
ビュー
Visual Studio
起動
ギャラリー
list505
favicon.ico
index.html
speech.1.0.0.js
index.html
<!DOCTYPE html>
<html lang="en">
<head>
localhost
localhost:53838
Start
[{"lexical":"hi","display":"Hi.","inverseNormalization":null,"maskedInverseNormalization":null,"transcript":"Hi.","confidence":0.7939022}]
100 %
ログ
説明
サイト
ファイル
データベース
レポート

図 5.41

(4) 音声で LED を制御してみる

BingSpeechAPI と IoTHub を組み合わせて BBG の LED を音声で点滅させてみます。デバイス側は温度計測＋LED 制御のフロー７を流用します。実行の様子を図 5.40、写真 5.1 に示します。

リスト８　server.js　/PC、Azure

・追加のライブラリ　'azure-iothub'　、'socket.io'

・managekey　は　Azure　「KEY1」

・リソース ID　は　Azure　「Propertise」の「RESOURCE ID」

・サブスクリプション ID は　Azure　「Propertise」の「SUBSCRIPTION ID」

```
var Client = require('azure-iothub').Client;
var connectionString
    = 'HostName=doilab.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=U9V/xxx';
var client = Client.fromConnectionString(connectionString);
var targetDevice = 'oi10dev';          // [Your Device ID]';

var http = require('http');
var fs = require('fs');
var server = http.createServer();
var sock = require('socket.io');
var io = sock.listen(server);
var gres;

// display output
function printResultFor(op) {
          return function printResult(err, res) {
                    if (err) console.log(op + ' error: ' + err.toString());
                    if (res) console.log(op + ' status: ' + res.constructor.name);
          };
}

server.on('request', function (req, res) {
    gres = res;
    console.log('on');
    console.log(req.url);

    if (req.url === '/') {
        fs.readFile("./index.html", "utf-8", errx);
        console.log('index');
    }
    if (req.url === '/speech.1.0.0.js') {
        fs.readFile("." + req.url, errx);
       console.log('speech');
    }
});
```

```
io.sockets.on('connection', function (socket) {
    socket.on('response', function ( data ) {
      console.log(data);
       socket.emit('echo', ' ¥nSet LED = ' + data);
       console.log('Sending message: ' + data);
          client.send(targetDevice, data, //message,
                                                printResultFor('send'));
    });
});

 function errx(err, data) {
      if (err) {
         gres.writeHead(404, {'Content-Type': 'text/plain'});
         gres.write('not found!');
         return gres.end();
      }else {
         gres.writeHead(200, {'Content-Type': 'text/html'});
         gres.write(data);
         gres.end();
      }
 }

//connect and send
client.open( function (err) {
          if (err) {
                    console.log('Could not connect: ' + err);
          } else {
                    console.log('Client connected');
          }
});

server.listen(process.env.PORT || 80);
console.log('Server listening...');
```

リスト8　index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">

    <script src="./speech.1.0.0.js" type="text/javascript"></script>
    <script src="/socket.io/socket.io.js"></script>

    <script>
        var socket = io.connect();

    </script>
    <script type="text/javascript">
        var client;
        var request;
        var rdata;

        function clearText() {
            document.getElementById("output").value = "";
        }

        function setText(text) {
            document.getElementById("output").value += text;
        }

        function start() {

            clearText();
            setText('Please talk( go , stop ) : ');
            client
            =
Microsoft.CognitiveServices.SpeechRecognition.SpeechRecognitionServiceFactory.createMicrophoneClientWithIntent(
                'en-us',                                // Language
                '80ab8d8952ca4ce4xxxx',          // key
'/subscriptions/8axxx/resourceGroups/doilabg/providers/Microsoft.CognitiveServices/accounts/csa00',
                                                // luisAppID
                '8a9aa7a7-ea34-4xxxx'            // luisSubID
                        );

            client.startMicAndRecognition();
            setTimeout(function () {
                client.endMicAndRecognition();
            }, 5000);

            client.onPartialResponseReceived = function (response) {
                setText(response);
            }
```

```
            client.onFinalResponseReceived = function (response) {
                var x = response[0]["lexical"];
                setText('¥nRespons : ' + x );

                if (x == "go") {
                    rdata = '1';
                    socket.emit('response', rdata);
                }
                else {
                    if (x == "stop") {
                        rdata = '0';
                        socket.emit('response', rdata);
                    }
                }
            }

            client.onIntentReceived = function (response) {
                setText(response);
            }
        }

        socket.on('echo', function (data) {
            setText(data);
        });
    </script>
</head>
<body style="font-family: Arial,Verdana, sans-serif">
    <table style="width:100%">
        <tr>
            <td><button onclick="start()">Start</button></td>
        </tr>
        <tr>
            <td>
                <textarea id="output" style='width:400px;height:200px'></textarea>
            </td>
        </tr>
    </table>
</body>
</html>
```

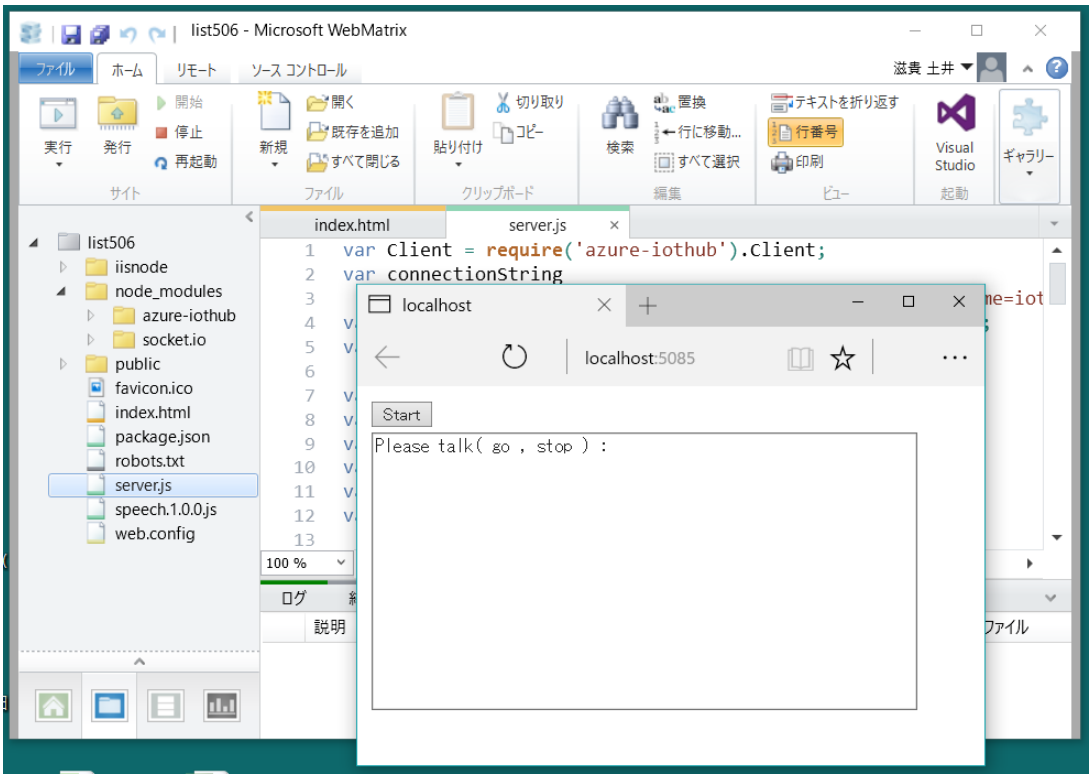
list506 - Microsoft WebMatrix
index.html
server.js
var Client = require('azure-iothub').Client;
var connectionString
localhost
localhost:5085
Start
Please talk(go , stop) :
list506
iisnode
node_modules
azure-iothub
socket.io
public
favicon.ico
index.html
package.json
robots.txt
server.js
speech.1.0.0.js
web.config

図 5. 42

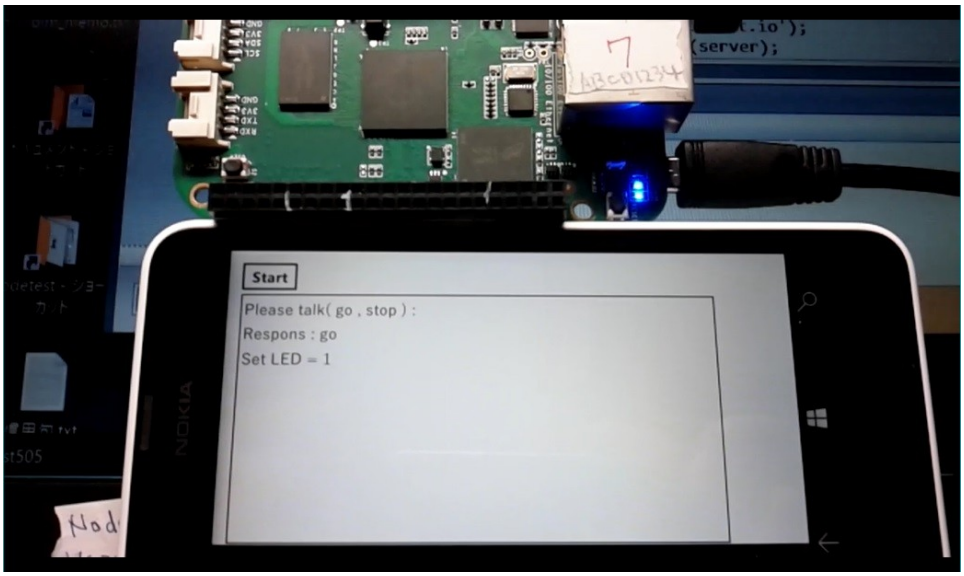
Start
Please talk(go , stop) :
Respons : go
Set LED = 1

写真 5. 1

5.4 Computer Vision API を試す

最後に画像系のサービスを試します。

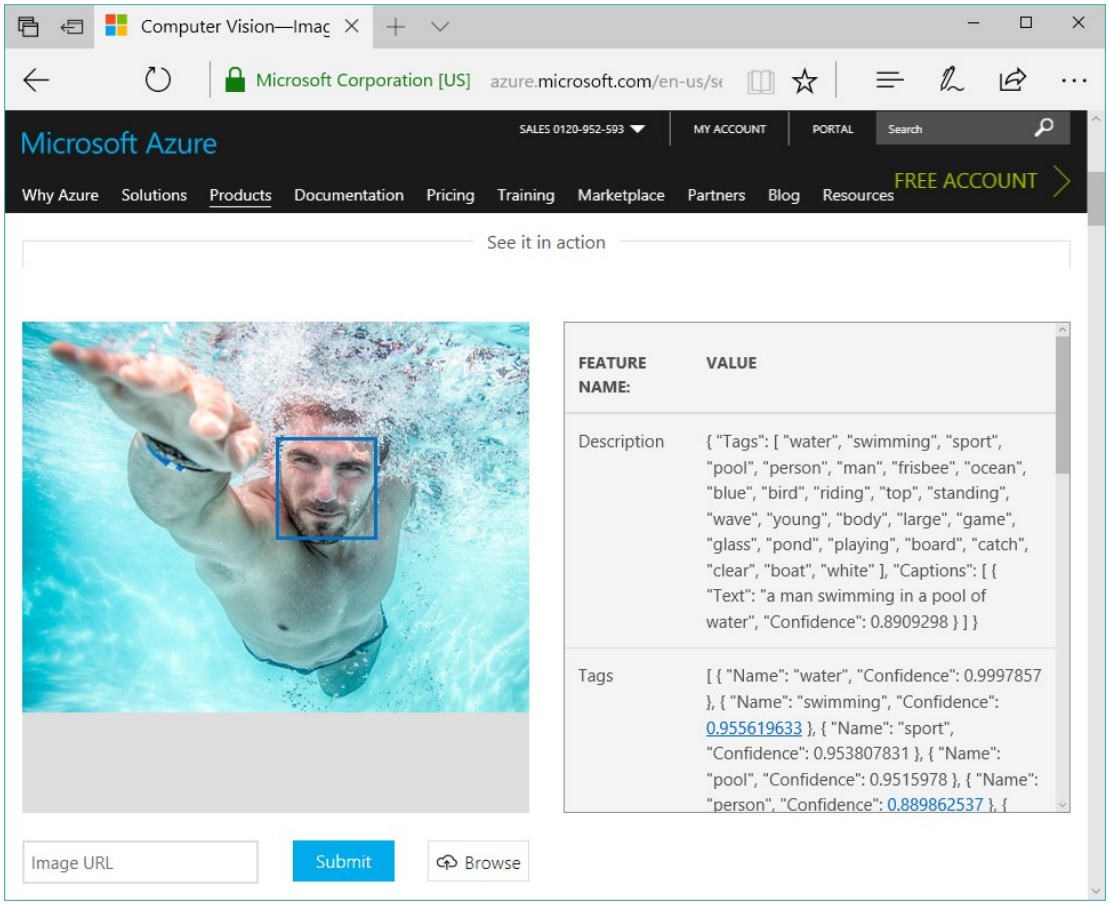

図 5.43　サンプルページ

https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/

(1)Node-RED で試してみる。

Azure 版の Node-RED にはいくつかの CognitiveService のノードが用意されています。今回はそのノードを使用します。使用する前に BingSpeechAPI と同様に Computer Vision API のサービスを作ります。その様子を図 5.44、図 5.45 に示します。そのあとフローを作ります。その様子を図 5.46 に示します。file inject ノードを使って処理したい画像ファイルを選択します。また認識の種類は Computer Vision ノードの設定で選択できます。その様子を図 5.47 に示します。

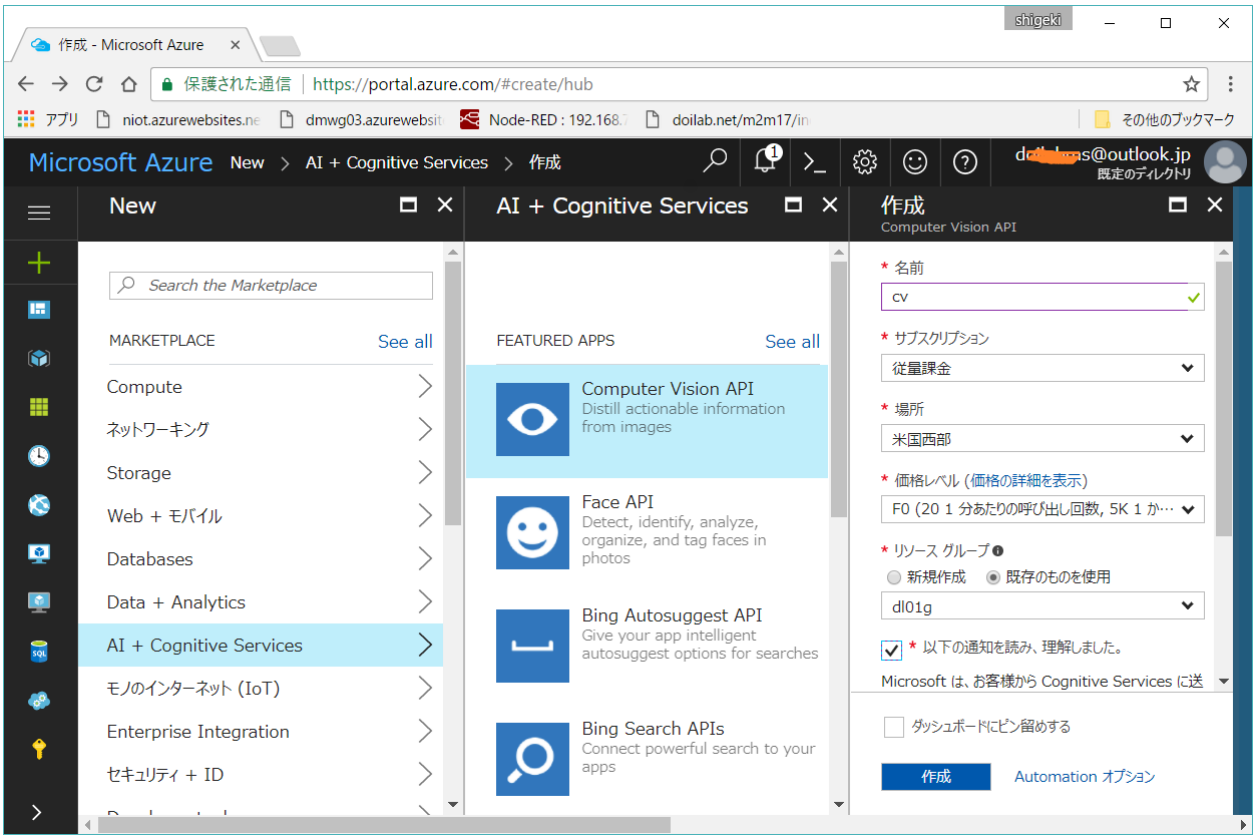
作成 - Microsoft Azure
https://portal.azure.com/#create/hub
Microsoft Azure
New
AI + Cognitive Services
作成
New
Search the Marketplace
MARKETPLACE
See all
Compute
ネットワーキング
Storage
Web + モバイル
Databases
Data + Analytics
AI + Cognitive Services
モノのインターネット (IoT)
Enterprise Integration
セキュリティ + ID
AI + Cognitive Services
FEATURED APPS
See all
Computer Vision API
Distill actionable information from images
Face API
Detect, identify, analyze, organize, and tag faces in photos
Bing Autosuggest API
Give your app intelligent autosuggest options for searches
Bing Search APIs
Connect powerful search to your apps
作成
Computer Vision API
* 名前
cv
* サブスクリプション
従量課金
* 場所
米国西部
* 価格レベル (価格の詳細を表示)
F0 (20 1 分あたりの呼び出し回数, 5K 1 か…
* リソース グループ
新規作成
既存のものを使用
dl01g
* 以下の通知を読み、理解しました。
Microsoft は、お客様から Cognitive Services に送
ダッシュボードにピン留めする
作成
Automation オプション

図 5.44

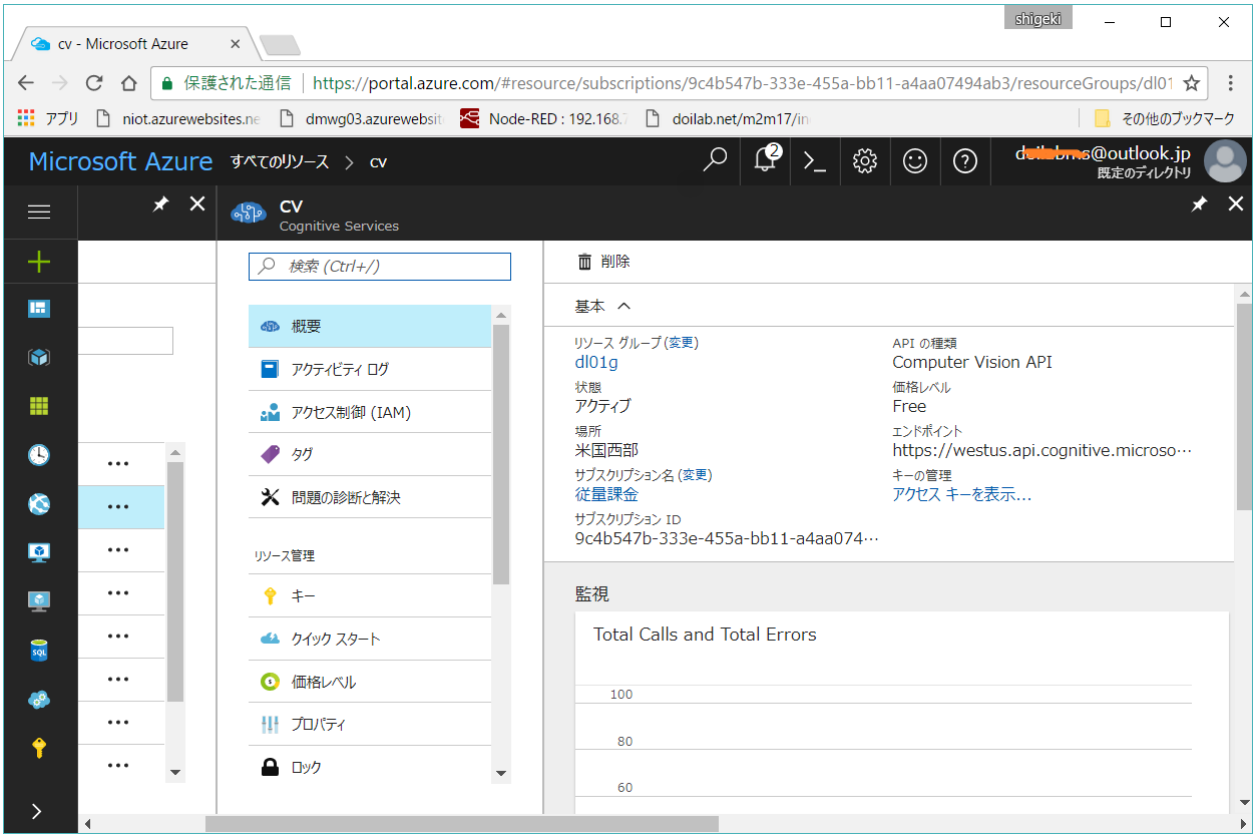
cv - Microsoft Azure
https://portal.azure.com/#resource/subscriptions/9c4b547b-333e-455a-bb11-a4aa07494ab3/resourceGroups/dl01
Microsoft Azure
すべてのリソース
cv
cv
Cognitive Services
検索 (Ctrl+/)
概要
アクティビティ ログ
アクセス制御 (IAM)
タグ
問題の診断と解決
リソース管理
キー
クイック スタート
価格レベル
プロパティ
ロック
削除
基本
リソース グループ (変更)
dl01g
状態
アクティブ
場所
米国西部
サブスクリプション名 (変更)
従量課金
サブスクリプション ID
9c4b547b-333e-455a-bb11-a4aa074…
API の種類
Computer Vision API
価格レベル
Free
エンドポイント
https://westus.api.cognitive.microso…
キーの管理
アクセス キーを表示…
監視
Total Calls and Total Errors
100
80
60

図 5.45

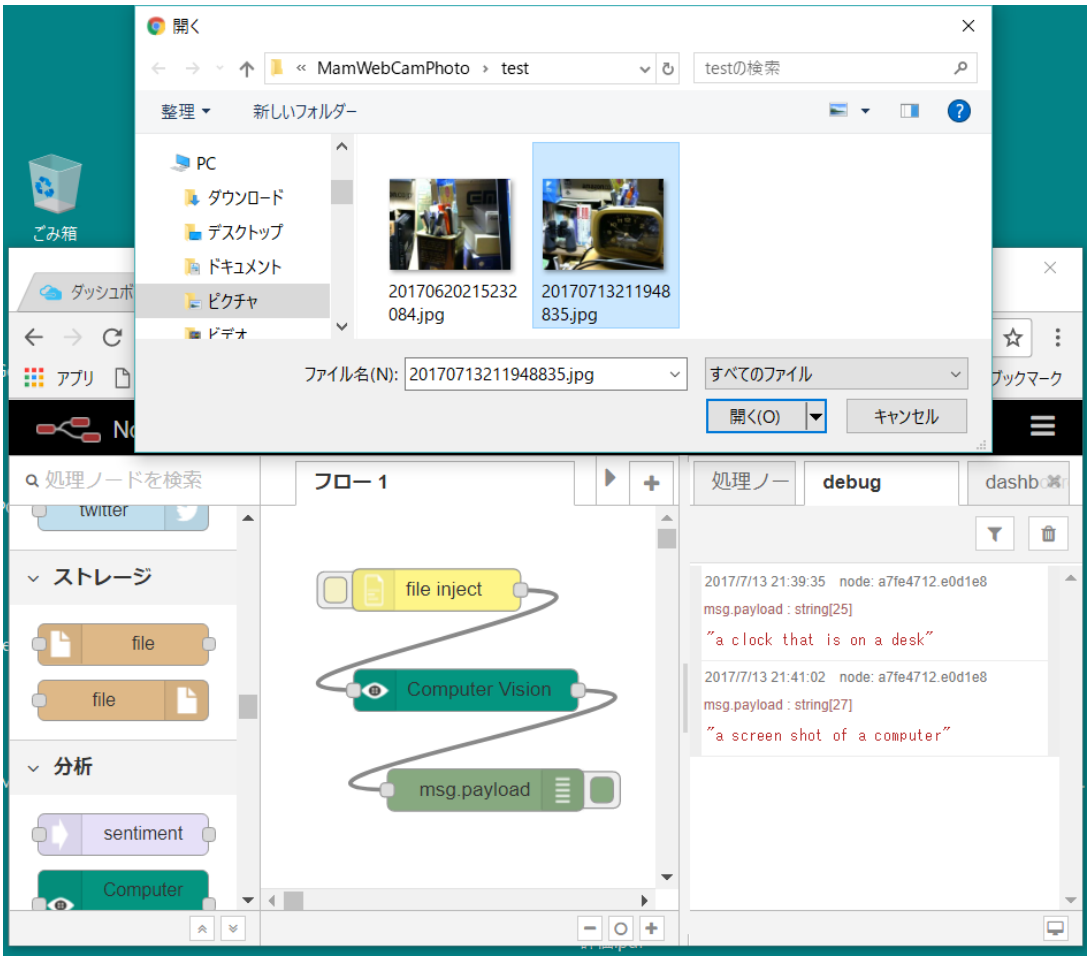
開く
MamWebCamPhoto › test
testの検索
整理
新しいフォルダー
PC
ダウンロード
デスクトップ
ドキュメント
ピクチャ
20170620215232084.jpg
20170713211948835.jpg
ファイル名(N): 20170713211948835.jpg
すべてのファイル
開く(O)
キャンセル
ごみ箱
処理ノードを検索
フロー 1
ストレージ
file
file
分析
sentiment
Computer
file inject
Computer Vision
msg.payload
処理ノー
debug
2017/7/13 21:39:35 node: a7fe4712.e0d1e8
msg.payload : string[25]
"a clock that is on a desk"
2017/7/13 21:41:02 node: a7fe4712.e0d1e8
msg.payload : string[27]
"a screen shot of a computer"

図 5.46

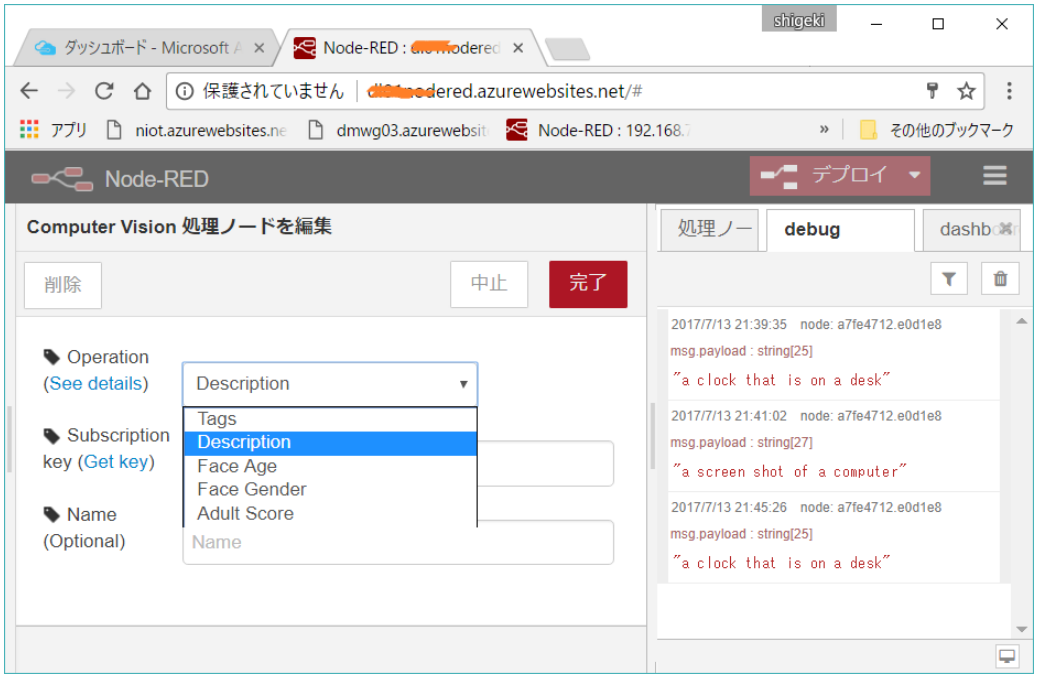
shigeki
Node-RED
デプロイ
Computer Vision 処理ノードを編集
削除
中止
完了
Operation
(See details)
Description
Tags
Description
Face Age
Face Gender
Adult Score
Subscription key (Get key)
Name (Optional)
Name
処理ノー
debug
2017/7/13 21:39:35 node: a7fe4712.e0d1e8
msg.payload : string[25]
"a clock that is on a desk"
2017/7/13 21:41:02 node: a7fe4712.e0d1e8
msg.payload : string[27]
"a screen shot of a computer"
2017/7/13 21:45:26 node: a7fe4712.e0d1e8
msg.payload : string[25]
"a clock that is on a desk"

図 5.47

まだ知らない人のための最新 Microsoft Azure 入門：
http://www.buildinsider.net/web/azure/01
Azure のサービス全体がよくまとまっている。

Node.js でテーブル storage を操作する解説：
https://docs.microsoft.com/ja-jp/azure/storage/storage-nodejs-how-to-use-table-storage

土井研究室、IoT のまとめページ：
http://doilab.net/web/jisyu/etc/iot/index.html
本講座の資料他。