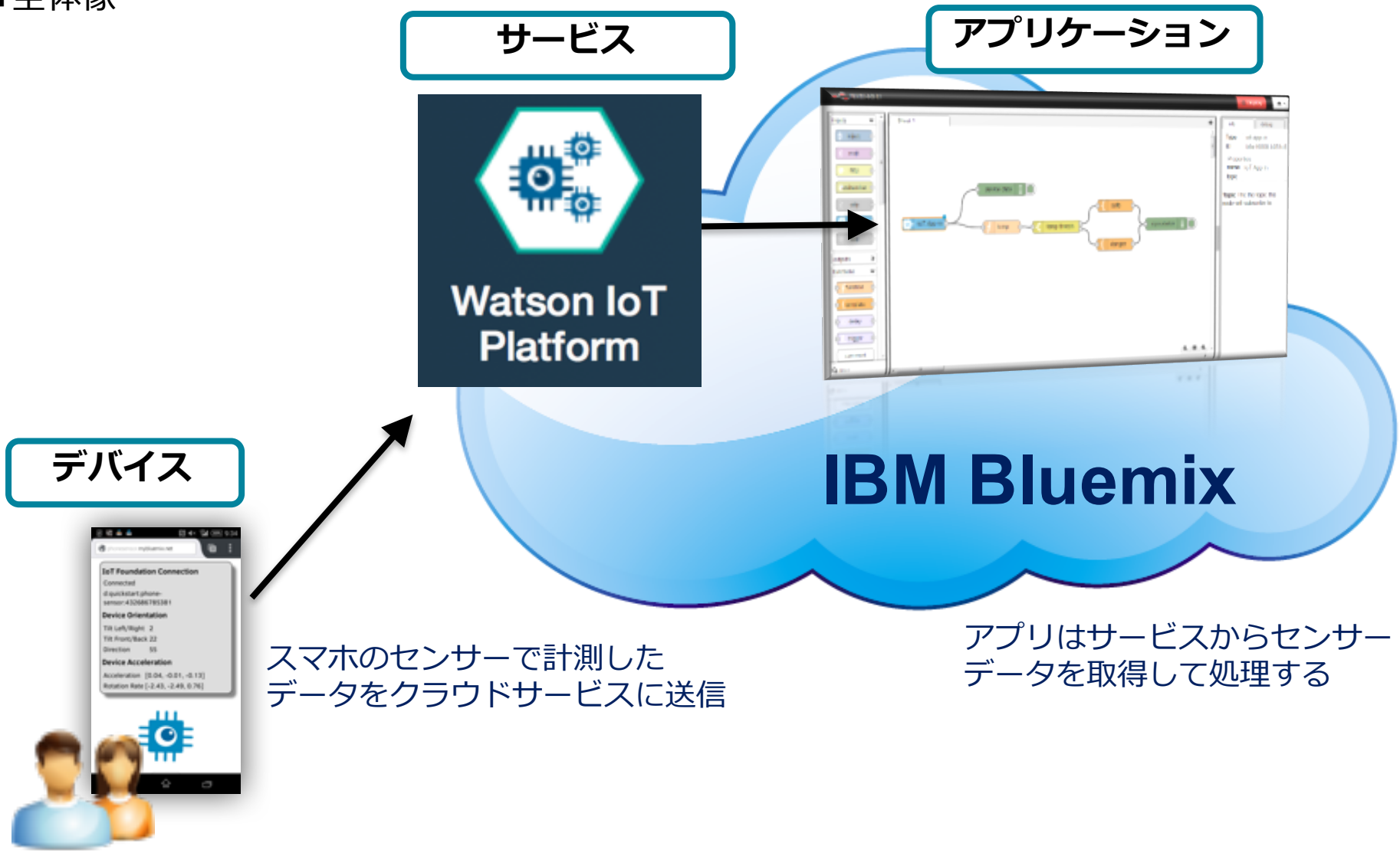
Bluemix Hands-On

Node-REDでIoT (Internet of Things) アプリを作成

■全体像



IoT アプリの作成

IoT (Internet of Things) アプリの作成

■Bluemixの米国のデータセンターを使用します。
　右上部の人のアイコンをクリックし、地域が「米国南部」であることを確認して下さい。

もし「スペースの作成」というウィンドウが
表示された場合は、任意の名前(dev等)を
指定してスペースを作成してください。

■上部メニューから「カタログ」をクリック。

■最上段の「ボイラープレート」カテゴリの中から
　「Internet of Things Platform Starter」
を選択。

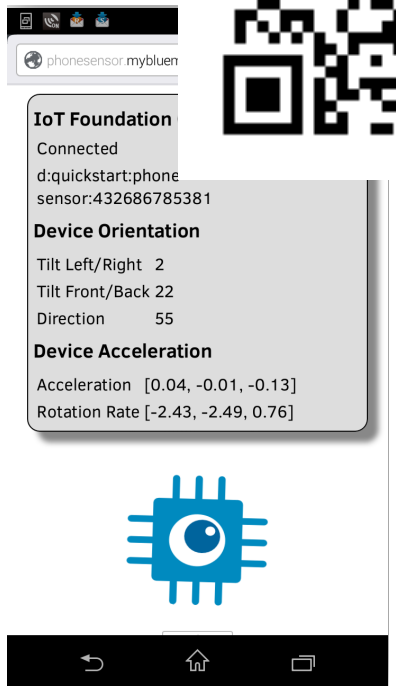
■アプリ名とホスト名を入力します。
実際にはアプリ名を入力フィールドに入れると、ホスト名も同じ名前が指定されます。
既に同じ名前のホストが存在する場合エラーとなりますので、他の人と重複しなそうな名前を指定してください。 例：server20160603xx など。

■「作成」ボタンをクリックすると、IoT Foundationの環境作成が始まります。
完了までしばらくお待ちください。

Node-REDでIoTアプリを作成

■スマートフォンをお持ちの方は、スマホのブラウザで下記URLにアクセスしてください。

http://phonesensor.mybluemix.net



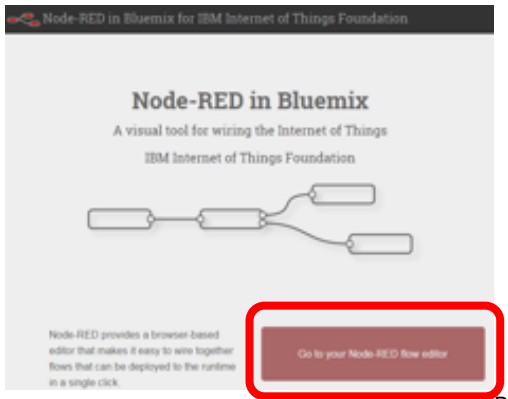
PhoneSensorが起動します。スマホに内蔵された各種センサーの情報を取得し、BluemixのIoT Platformサービスに送信しています。

■このセンサー情報を受け取って処理するアプリをBluemix上のNode-REDで作ってみましょう。

Bluemixのダッシュボードから、先程作成したボイラープレートのアプリのURLを開き、Node-REDを起動します。

直接Webブラウザで URLを指定して開いてもOKです。

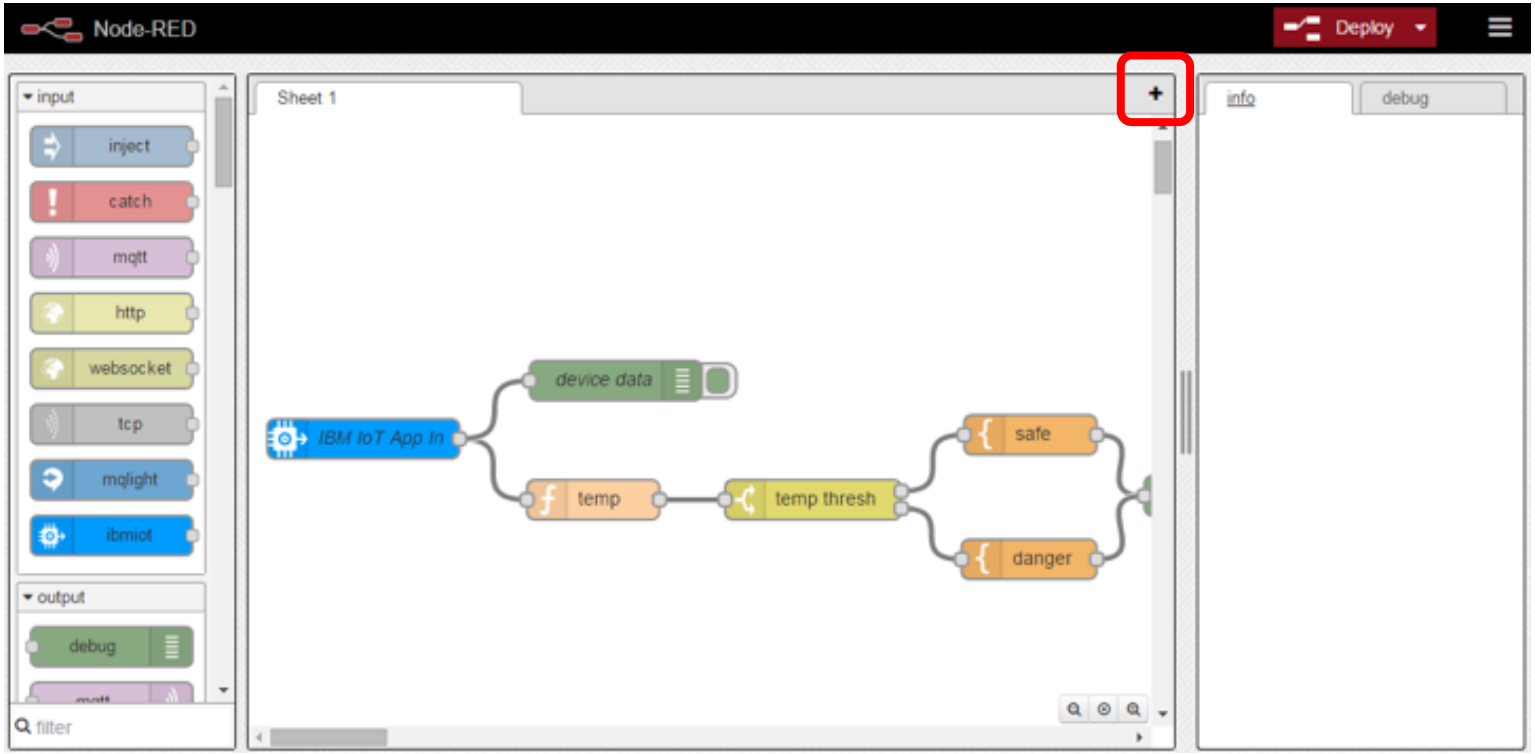
右の例の場合はこちらのURLです。
http://IoTServer2015xx.mybluemix.net



Node-REDでIoTアプリを作成

■Node-REDが起動します。

Sheet 1にサンプルのフローが用意されていますが、これは使用しないので別シートを追加します。
上部の「+」（プラス）ボタンをクリックします。



■ Sheet 2が開きますので、ここで処理フローを作成してみましょう。

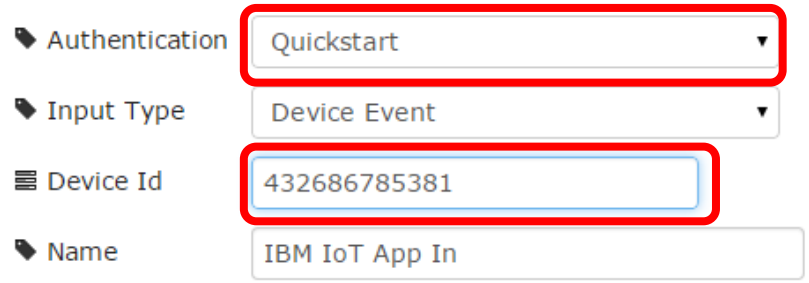
Node-REDでIoTアプリを作成

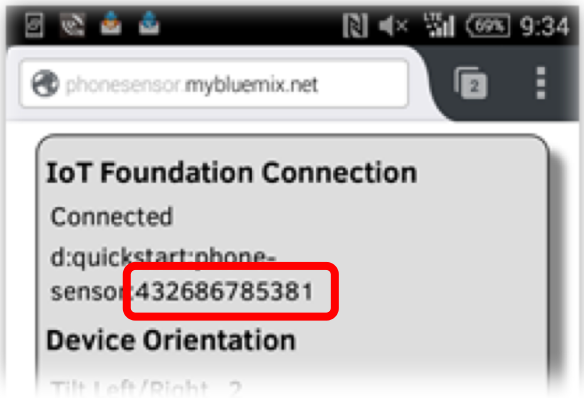
■スマホのセンサー情報を、クラウド上のIoT Foundation サービスから受け取ってみましょう。

左側のパレットから「ibmiot」ノードを中央のキャンバスにドラッグ＆ドロップします。

■キャンバスにドロップした「IBM IoT 」ノードをダブルクリックして設定画面を開きます。

Authentication は「Quickstart」を選択。
Device Id に先程のPhoneSensorに表示されている文字列の末尾12桁の数字を入力。

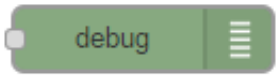


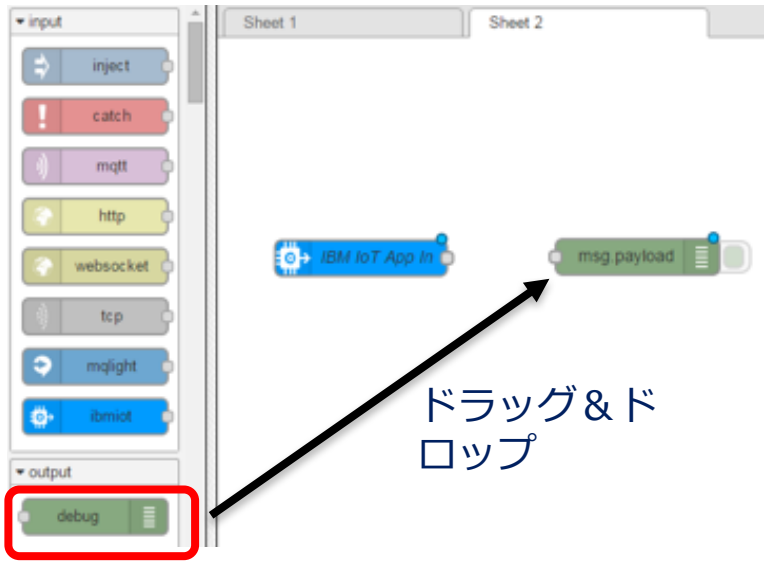


Node-REDでIoTアプリを作成

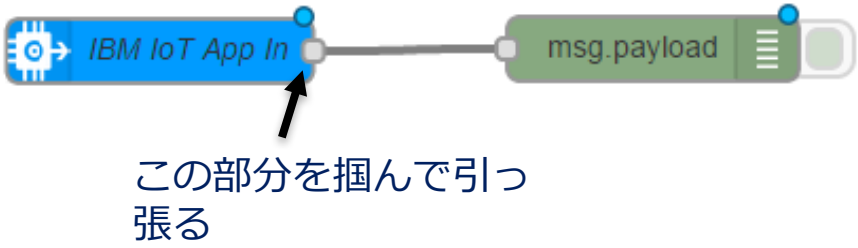
■受け取ったデータを表示するノードを用意します。

左側のパレットから「debug」ノードを中央のキャンバスにドラッグ＆ドロップします。





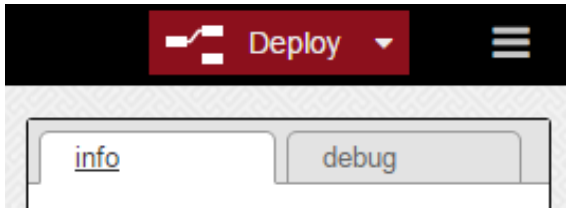
■ノードの横にあるコネクタをクリック＆ホールドし、「ibmiot」ノードと「debug」ノードを線で繋ぎます。



Node-REDでIoTアプリを作成

■これでデータの入力と出力の定義ができました。

それでは実際に動かしてみましょう。
右上の「Deploy」ボタンをクリックします。



■画面上部に「Successfully Deployed」の文字が表示されればOKです。
（この文字はすぐに消えます）

■実行結果を確認してみましょう。
右上の「debug」タブを選択します。debugノードに流れてきたデータはここに表示されます。

先程のPhoneSensorを動かします。スマホのセンサー情報がdebug画面に表示されれば成功です。

取得できる情報はスマホの機種によって
異なる場合があります。

Node-REDでIoTアプリを作成

■debug画面にセンサーデータがうまく表示されない場合のチェックポイント

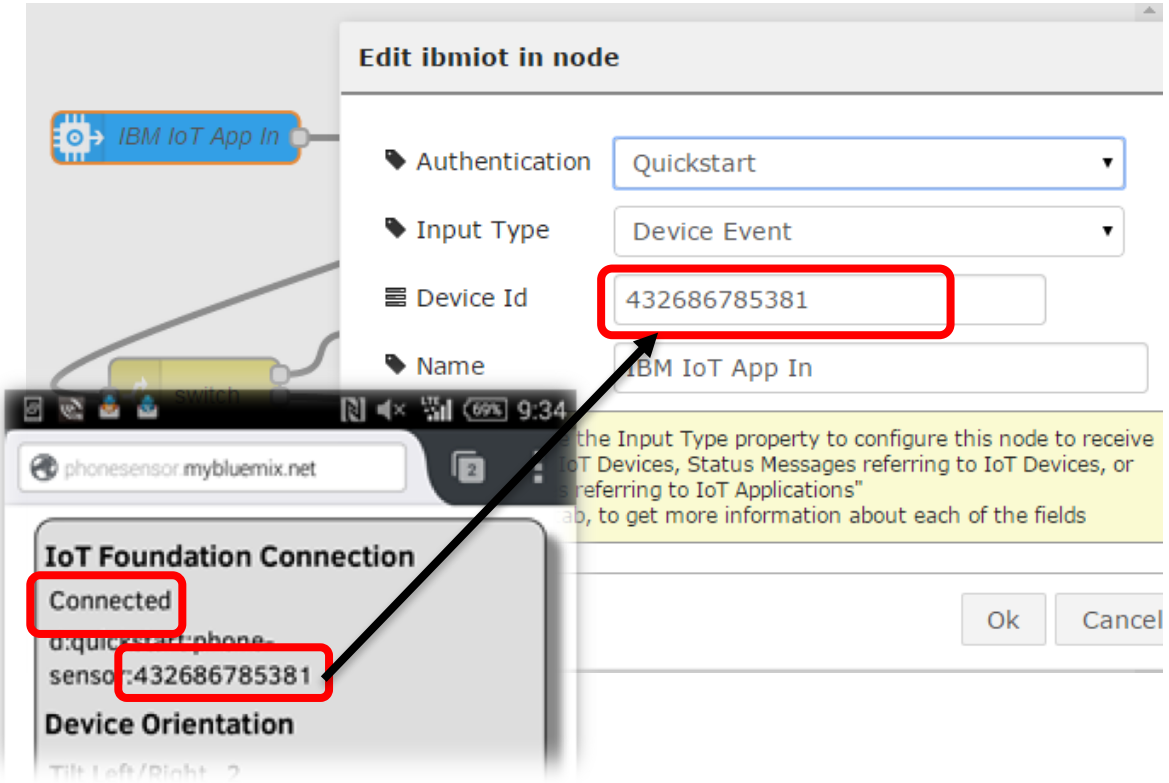
-PhoneSensorが正しく稼動しているかどうかを確認してください。スリープ状態でアプリが
停止していないでしょうか。

-PhoneSensorの接続ステータスが「Connected」になっていることを確認してください。
例えば、誤ってPhoneSensor画面下部の「Register」をタップしてしまうと、ステータスが
「Disconnected」になります。

-「ibmiot」ノードのDevice Idが
正しく設定されているかどうかを
ご確認ください。

例えば、スマホのWebブラウザで
PhoneSensorを再読み込みすると
Device Idが変わってしまいます。

その場合は再度「ibmiot」ノードの
定義情報をPhoneSensor側と
同じになるよう変更してください。



Node-REDでIoTアプリを作成

■正しくセンサー情報が取得できたので、この情報に応じて何らかの処理を行うフローを作ってみましょう。

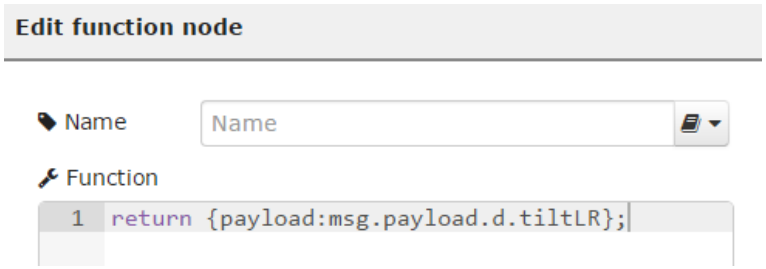
「tiltLR」というデータ項目に着目します。
これはスマホ端末の左右の傾きを測定しているようです。
実際に端末を動かして、値の変化を確認してください。

この傾き度合いによって表示メッセージを変えてみましょう。

■センサー情報から「tiltLR」の値だけを抽出します。

左側のパレットから「function」ノードを中央のキャンバスにドラッグ＆ドロップします。

ダブルクリックで設定画面を開き、Functionの内容を以下のように書き換えて「OK」をクリックします。



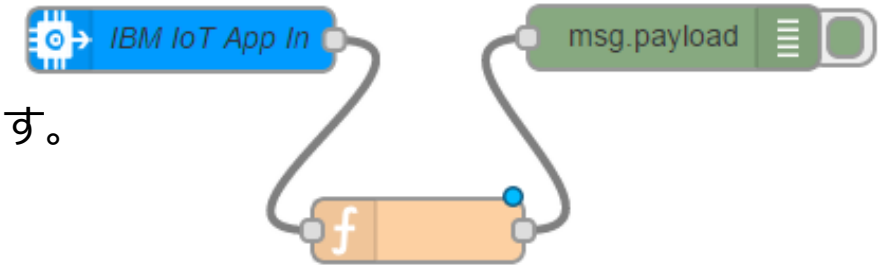

```
return {payload:msg.payload.d.tiltLR};
```

Node-REDでIoTアプリを作成

■正しく動くかどうか確認してみましょう。

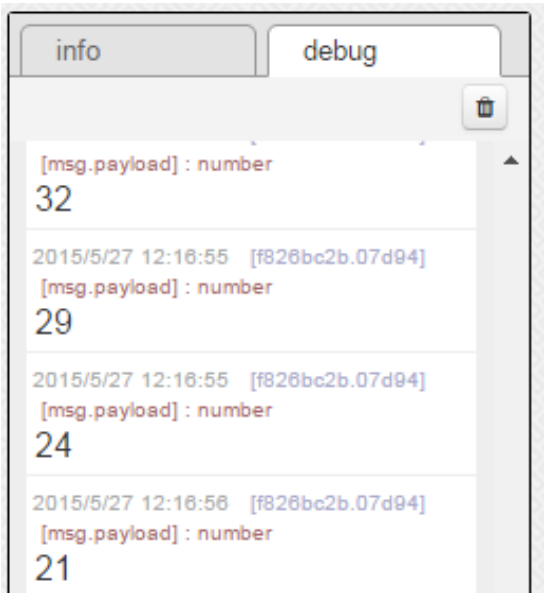
「ibmiot」と「debug」との間に、今作成した
「function」ノードを挟むように線を繋ぎ変えます。

線はクリックで選択した状態でDeleteキーで
削除できます。



■右上の「Deploy」ボタンをクリックします。

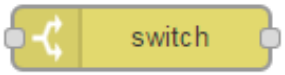
■不要な情報が削除されて、「tiltLR」の値だけが
表示されればOKです。



Node-REDでIoTアプリを作成

■「tiltLR」の値によって処理を分岐させます。

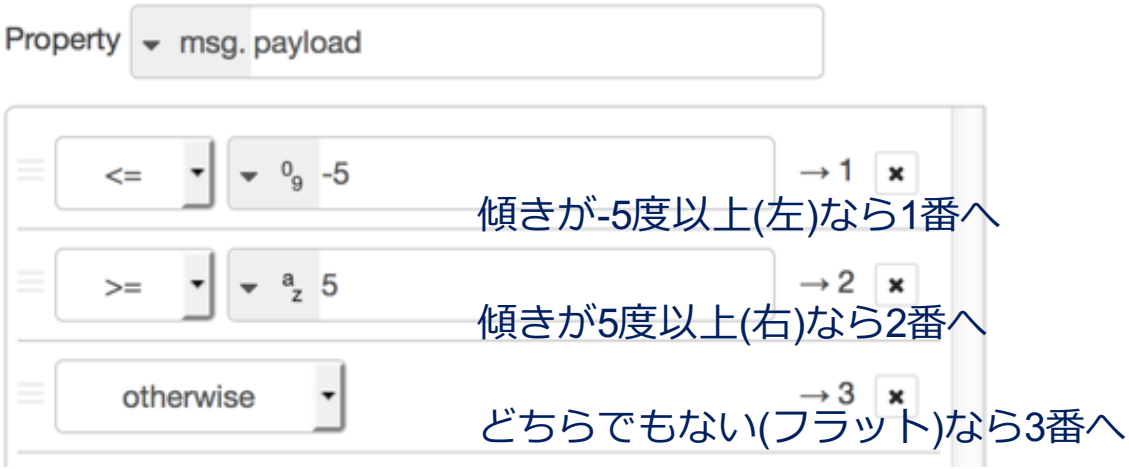
左側のパレットから「switch」ノードを中央のキャンバスにドラッグ＆ドロップします。



ダブルクリックで設定画面を開き、分岐条件を指定します。

左に5度以上傾いた場合＝「左に傾いてます」
右に5度以上傾いた場合＝「右に傾いてます」
傾きが5度未満の場合＝「ほぼフラットです」

上記のような条件設定をするには右図のように指定して「OK」をクリックします。



条件を追加するには、「+rule」ボタンをクリックします。

Node-REDでIoTアプリを作成

■条件分岐された後の処理を作成します。

左側のパレットから「template」ノードを中央のキャンバスに
ドラッグ＆ドロップします。

ダブルクリックで設定画面を開き、表示するメッセージを定義します。

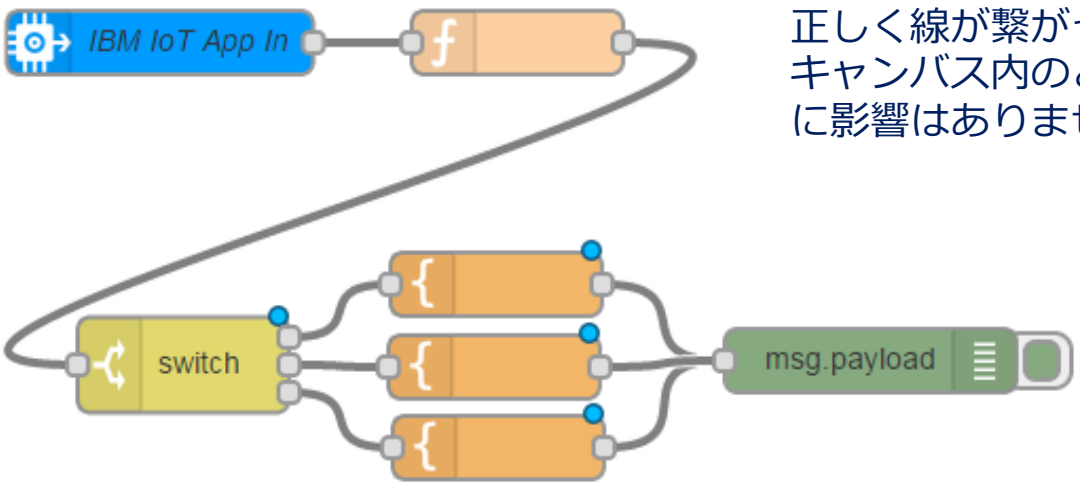
ご自由にメッセージの内容を入力してください。

{{payload}} と指定すると、先程
debug画面で確認した傾きデータ
の値が表示されます。

同様に2番、3番のノードも作成し、
それぞれメッセージを定義します。
最後にswitchノードのコネクタと
1番、2番、3番のノードを線で繋げます。

Node-REDでIoTアプリを作成

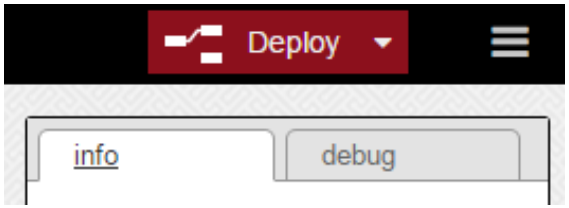
■キャンバスの中を整理して、各ノードを以下の図のように接続してください。



正しく線が繋がっていれば、ノード自体はキャンバス内のどこに置かれていても動作に影響はありません。

■処理フローが完成しました！

それでは実際に動かしてみましょう。
右上の「Deploy」ボタンをクリックします。



Node-REDでIoTアプリを作成

■「Successfully Deployed」のメッセージが表示されれば成功です。

PhoneSensorを稼動させたスマホを左右に傾けて
指定したメッセージが表示されることを確認してください。

お疲れ様でした！

IBM Bluemix と Node-RED によって、ほぼノンプログラミングで
IoTデバイスから収集したデータを処理するアプリが作れました。

Web APIを活用したアプリを動かす

Node-REDでWeb APIを活用したアプリを動かす

■Node-REDは作成したフローをJSON形式の文字列としてexport/importすることができます。

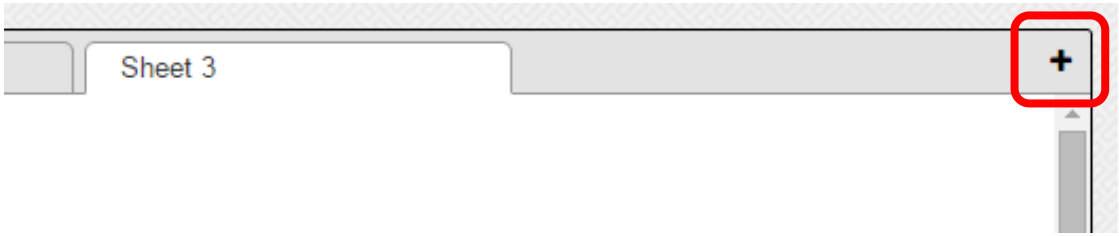
■あらかじめ作成済みの処理フローをde-REDに取り込んでみましょう。
サンプルアプリのテキストファイルをメモ帳で開き、文字列をすべて選択→コピーします。

テキストファイルの配布については講師よりガイドいたします。

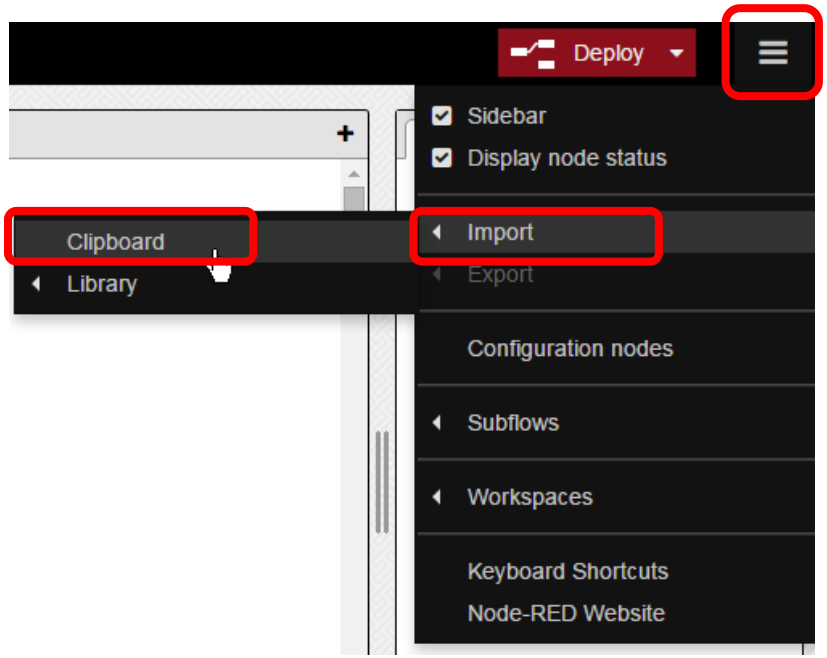

```
[{"id":"50e9e66b.e7938","type":"websocket-listener","path":"/ws/stations","wholemsg":"false"},{"id":"606e214e.3dc44","type":"websocket
out","z":"50b37b97.0d5ac4","name":"","server":"50e9e66b.e7938","x":636,"y":161,"wires":[]},{"id":"38a07a06.55b54e","type":"xml","z":"50b37b97.0d5ac4","name":"","
attr":"$","chr":"_","x":457.5,"y":162,"wires":[["606e214e.3dc44","8c5efd66.4718c8"]]},{"id":"8c5efd66.4718c8","type":"debug","z":"50b37b97.0d5ac4","name":"","active
":true,"console":"false","complete":"false","x":585.5,"y":227,"wires":[]},{"id":"db955c6b.60a0e8","type":"http request","z":"50b37b97.0d5ac4","name":"リクナビAPI呼
び出し
","method":"GET","url":"http://webservice.recruit.co.jp/shingaku/school/v2/?code=SC000332&key=43ba47966ce3abea","x":285,"y":162,"wires":[["38a07a06.55b54e"]
]},{"id":"f88a1161.b5d13","type":"websocket
in","z":"50b37b97.0d5ac4","name":"","server":"50e9e66b.e7938","x":85,"y":162,"wires":[["db955c6b.60a0e8"]]},{"id":"34e28e67.16041a","type":"http
response","z":"50b37b97.0d5ac4","name":"","x":451,"y":68,"wires":[]},{"id":"76e16db1.dae23c","type":"template","z":"50b37b97.0d5ac4","name":"表示内容の定義
","field":"payload","fieldType":"msg","syntax":"mustache","template":"<!DOCTYPE html>¥n<html>¥n <head>¥n <meta name=¥"viewport¥" content=¥"initial-
scale=1.0, user-scalable=no¥">¥n <meta charset=¥"utf-8¥">¥n <title>成城大学はここ！</title>¥n <style>¥n html, body, #map-canvas {¥n height:
94%;¥n margin: 5px;¥n padding: 0px¥n }¥n </style>¥n <script src=¥"https://maps.googleapis.com/maps/api/js?v=3.exp¥"></script>¥n
<script>¥n¥t¥tfunction initialize() {¥n¥t¥t var myLatlng = new google.maps.LatLng(35.65, 139.6);¥n¥t¥t var mapOptions = {¥n¥t¥t zoom: 13,¥n¥t¥t center:
myLatlng¥n¥t¥t };¥n¥t¥t var map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);¥n¥t¥t ¥n¥t¥t var loc = window.location;¥n
if (loc.protocol === ¥"https:¥") {¥n newUri = ¥"wss:¥";¥n } else {¥n newUri = ¥"ws:¥";¥n }¥n newUri += ¥"//¥" + loc.host +
¥"/ws/stations¥";¥n¥t¥t ¥n¥t¥t ¥n¥t¥t var sock = new WebSocket(newUri);¥n¥t¥t sock.onopen = function(){ ¥n¥t¥t console.log(¥"Connected
websocket¥");¥n¥t¥t console.log(¥"Sending ping..¥");¥n¥t¥t¥tsock.send(¥"Ping!¥");¥n¥t¥t console.log(¥"Ping sent..¥");¥n¥t¥t };¥n¥t¥t sock.onerror =
function(){ console.log(¥"Websocket error¥"); };¥n¥t¥t sock.onmessage = function(evt){¥n¥t¥t var schoolData = JSON.parse(evt.data);¥n¥t¥t var stations =
schoolData.results.school[0].campus;¥n¥t¥t¥tfor(var i = 0; i < stations.length; i++) {¥n¥t¥t¥t var camp = stations[i];¥n var marker = new
google.maps.Marker({¥n¥t¥t¥t position: new google.maps.LatLng(camp.latitude[0],camp.longitude[0]),¥n¥t¥t¥t map: map,¥n¥t¥t¥t animation:
google.maps.Animation.DROP,¥n¥t¥t¥t title: camp.address[0]¥n¥t¥t¥t });¥n¥t¥t¥t
¥n¥t¥t¥t}¥n¥t¥t };¥n¥t¥t};¥n¥t¥t¥n¥t¥tgoogle.maps.event.addDomListener(window, 'load', initialize);¥n¥n </script>¥n </head>¥n <body>¥n <div id=¥"map-
canvas¥"></div>¥n <font size=¥"6¥">成城大学はココ！</font>¥n <br>powered by IBM Bluemix and Node-RED¥n <div align=¥"right¥"> ¥n <a
href=¥"http://webservice.recruit.co.jp/¥"><img src=¥"http://webservice.recruit.co.jp/banner/shingaku-m.gif¥" alt=¥"リクナビ進学 Webサービス¥" width=¥"88¥"
height=¥"35¥" border=¥"0¥" title=¥"リクナビ進学 Webサービス¥"></a>¥n </div> ¥n
</body>¥n</html>","x":289,"y":68,"wires":[["34e28e67.16041a"]]},{"id":"3d94cc3b.6c46fc","type":"http
in","z":"50b37b97.0d5ac4","name":"","url":"/schoolmap","method":"get","x":102,"y":68,"wires":[["76e16db1.dae23c"]]}]
```

Node-REDでWeb APIを活用したアプリを動かす

■Node-REDでシートを追加します。
　上部の「+」（プラス）ボタンをクリックして、Sheet 3 を開きます。

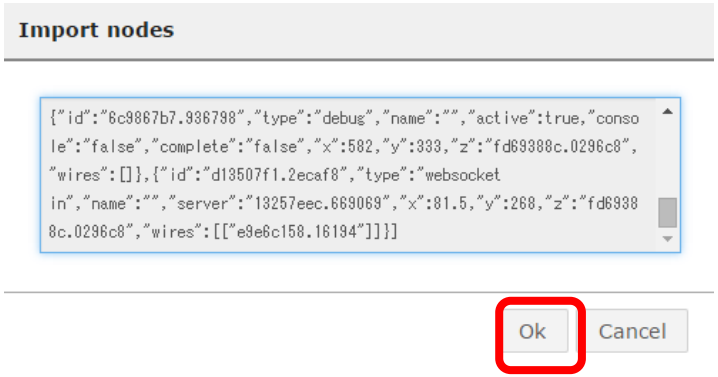


■Sheet 3 に先程コピーした処理フローを取り込みます。
　上部右端の三本線のアイコンをクリックし、表示されたメニューから「import」→「Clipboard」を
　選択します。



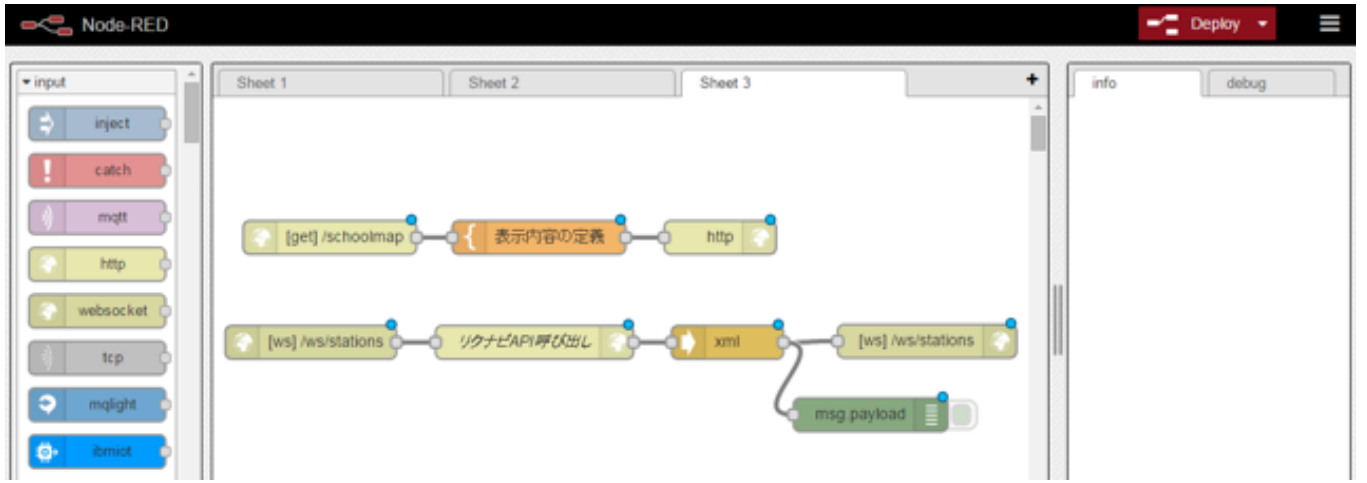
Node-REDでWeb APIを活用したアプリを動かす

■「Import nodes」ウィンドウが表示されるので、先程クリップボードにコピーしたテキストをペーストして、OKボタンをクリックします。



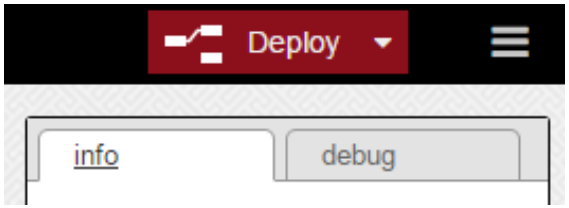
■処理フローが取り込まれますので、キャンバスの適当な場所をクリックして配置します。

■このアプリを動かしてみましょう。
右上の「Deploy」ボタンをクリックします。



Node-REDでWeb APIを活用したアプリを動かす

■このアプリを動かしてみましょう。
　右上の「Deploy」ボタンをクリックします。



■画面上部に「Successfully Deployed」の文字が表示されればOKです。

■実行結果を確認してみましょう。

　Webブラウザの新しいタブを開いて、Node-REDのホスト名の末尾に「/schoolmap」という
　パスを付加したURLにアクセスします。

(例)　http://iotserver2015xx.mybluemix.net/red/#　でNode-REDを開いている場合、

↓

http://iotserver2015xx.mybluemix.net/schoolmap　にアクセスします。

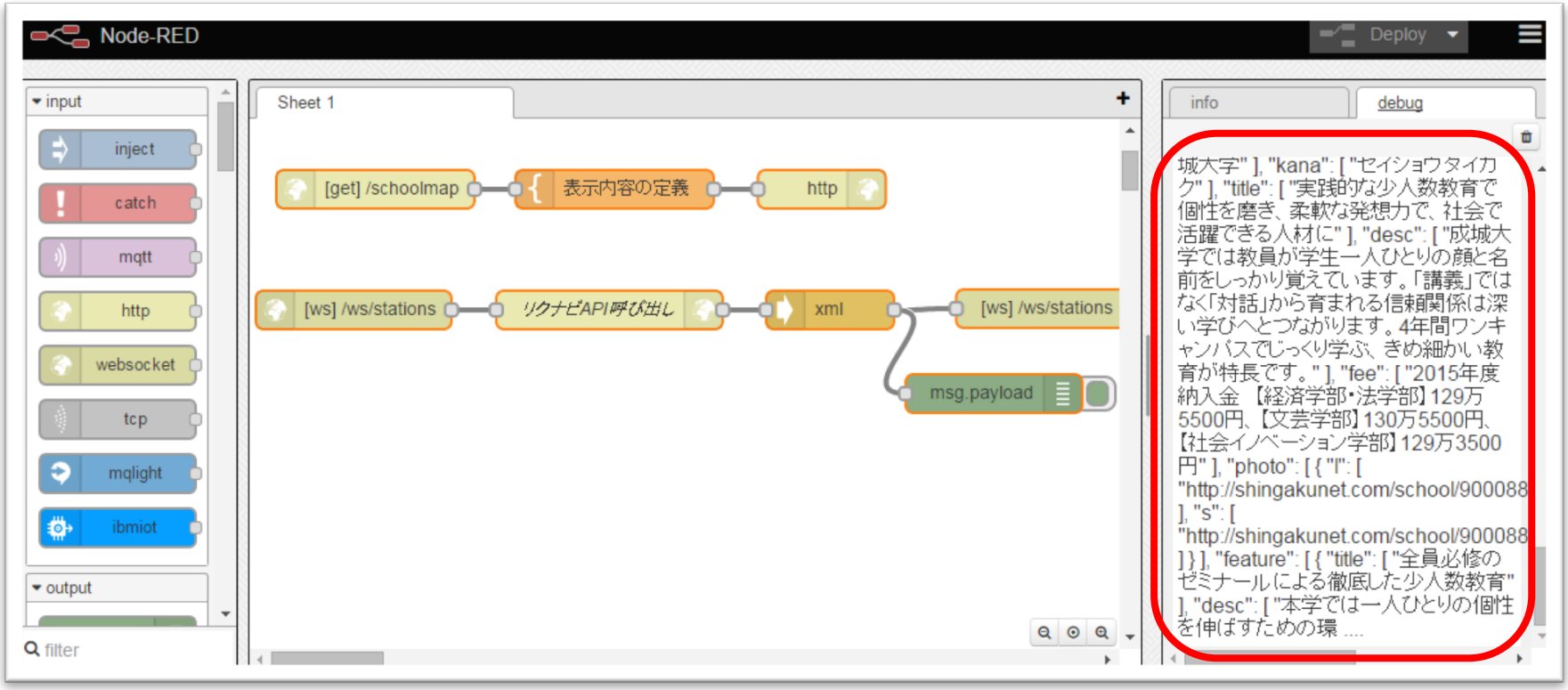
Node-REDでWeb APIを活用したアプリを動かす

■Node-REDで作ったサンプルアプリ「成城大学はココ！」が表示されればOKです。

地図上の成城大学のキャンパスにマーカーが表示され、マウスポインタを合わせると大学の住所が表示されます。

Node-REDでWeb APIを活用したアプリを動かす

■debug画面に、リクナビ進学APIを呼び出した結果が表示されていることを確認してください。
　　Webブラウザに表示していないものも含め、様々な情報が取得できています。

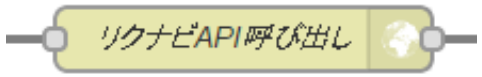


このようにクラウドで公開されている外部のAPIを呼び出すことで、アプリの機能を拡張できます

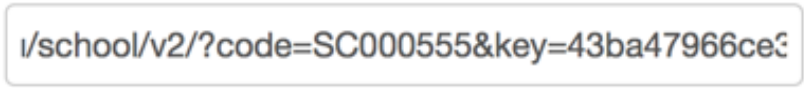
Node-REDでWeb APIを活用したアプリを動かす

■試してみましょう
　リクナビ進学APIに対してリクエストする内容を変えてみましょう。

■「リクナビAPI呼び出し」ノードをダブルクリックして構成画面を開きます。



■URL欄の「keyword=成城大学」の部分を「code=SC000555」と変更して「OK」をクリック。

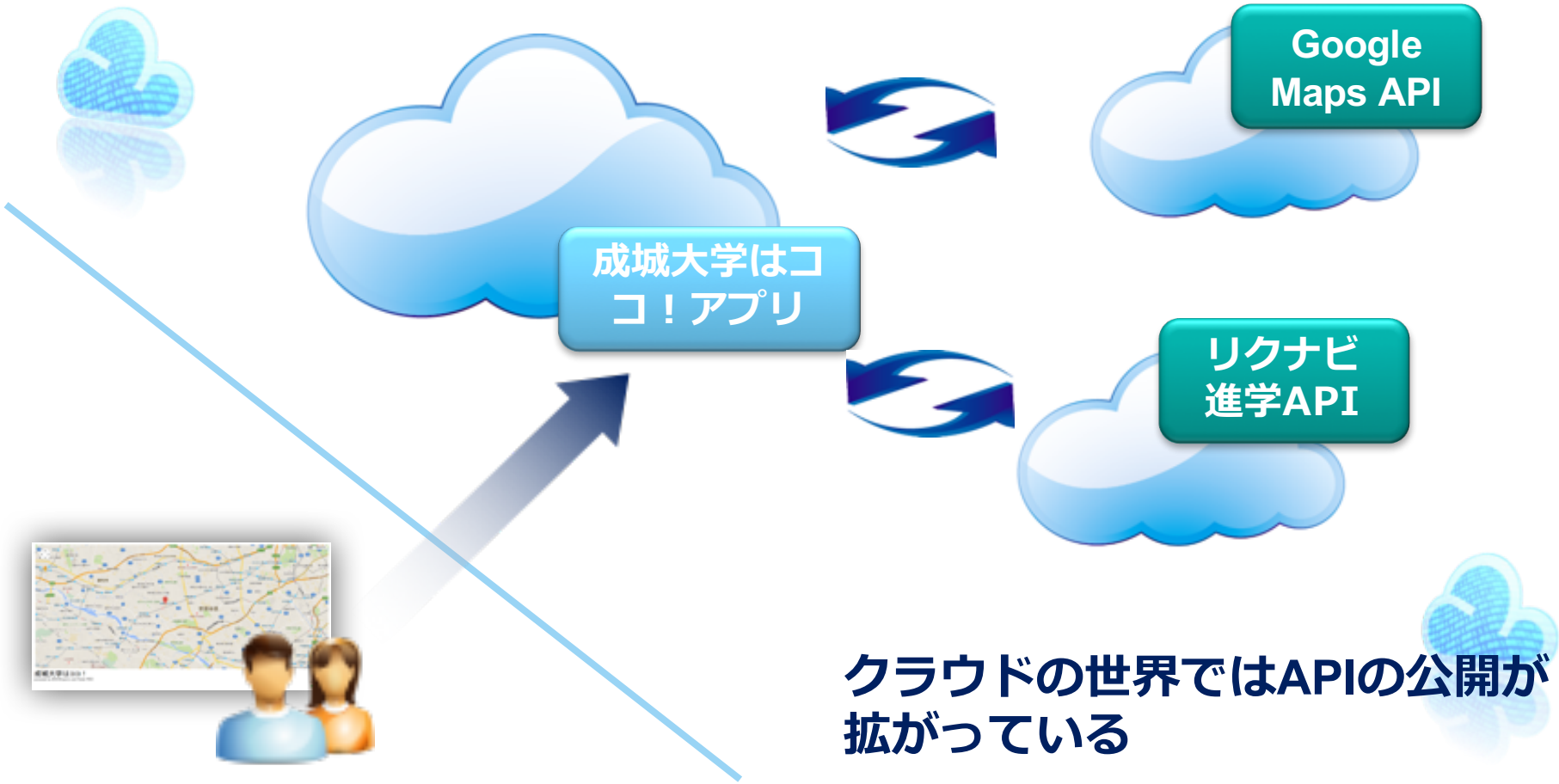


■デプロイして結果を見てみましょう。
　右上の「Deploy」ボタンをクリックします。
　「成城大学はココ！」をWebブラウザで再読み込みすると、何が変わるでしょうか？

API活用によるアプリ開発とIBM Bluemix

■「成城大学はココ！」アプリはBluemix環境のNode-RED上で稼動しています。

地図はGoogle Maps APIを、大学の住所および緯度経度の情報はリクナビ進学APIを呼び出して統合し、Webブラウザ上に表示しています。



クラウドの世界ではAPIの公開が拡がっている