

# Bluemix Hands-On #1

Kifumi Numata  
University Relations, IBM Japan



# 本日のハンズオン

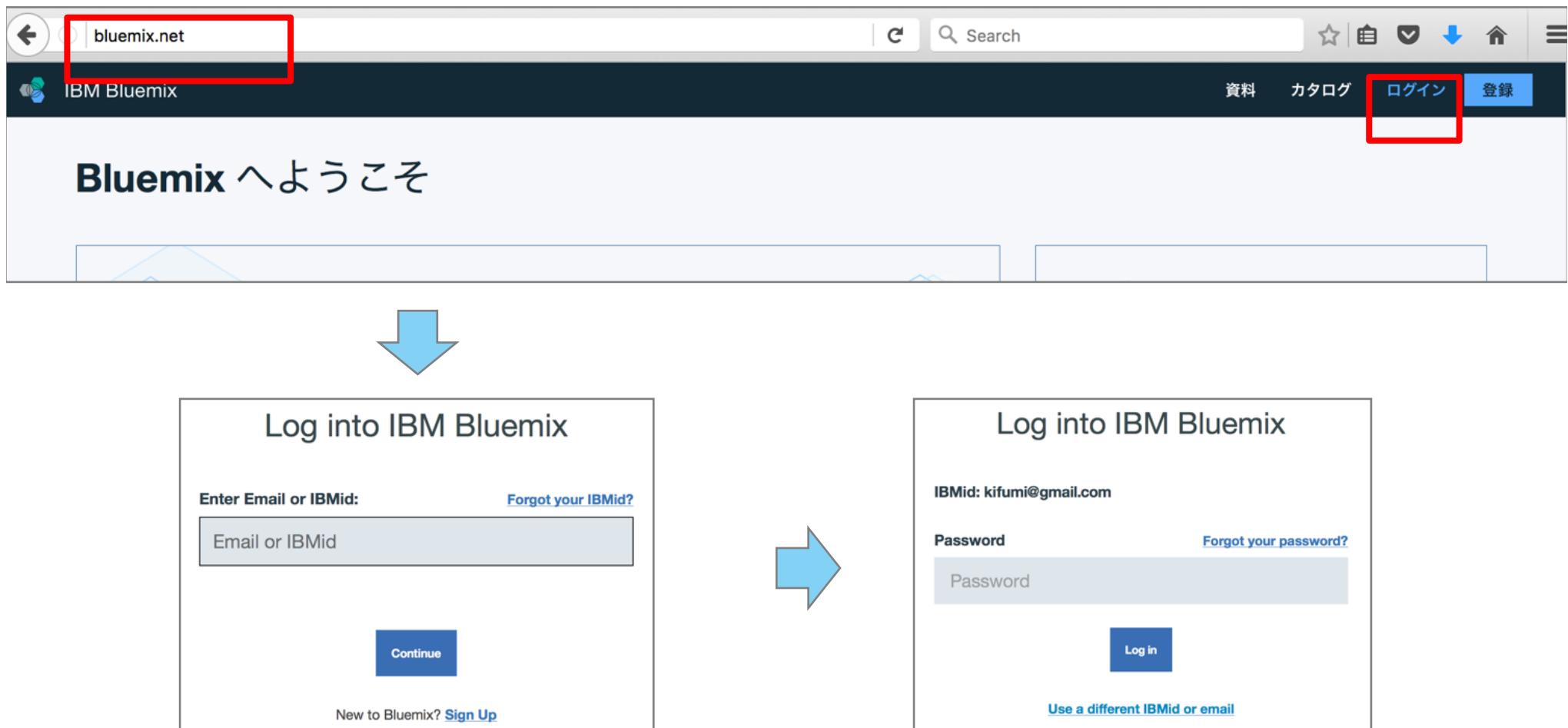
- Bluemixへのログイン
- Node-REDでHello World!
- IBM Watsonとは
- Watson Visual Recognition APIを使った画像認識アプリの作成

# 本日のハンズオン

- Bluemixへのログイン
- Node-REDでHello World!
- IBM Watsonとは
- Watson Visual Recognition APIを使った画像認識アプリの作成

# Bluemixへのログイン

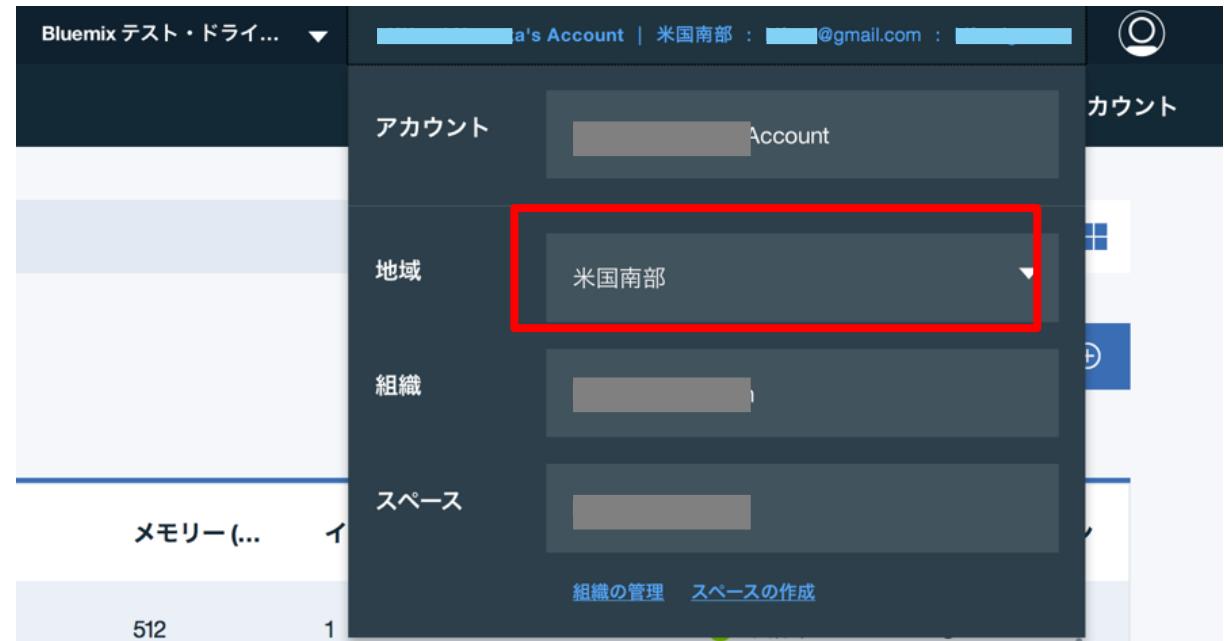
1. ブラウザーは、Firefox か Chromeを使って下さい。
2. bluemix.net にアクセスします。
3. 「ログイン」をクリック。
4. IBM ID (メールアドレス)とパスワードを入力して下さい。



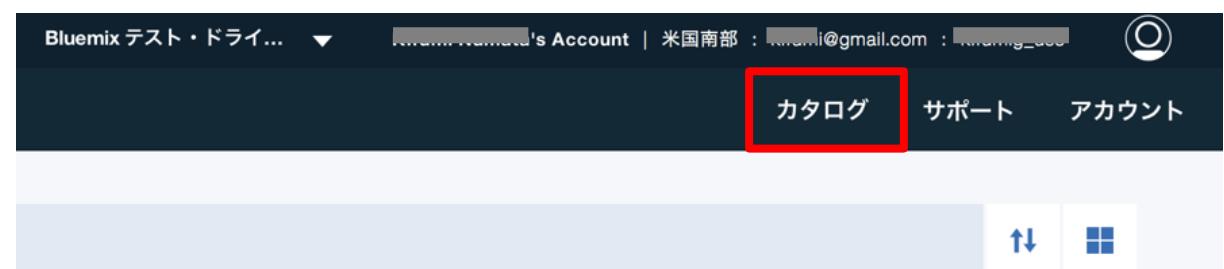
# Node-RED環境の作成

- 今回は、Bluemixの米国のデータセンターを使用することを想定します。  
右上のメールアドレスのあたりをクリックし、「米国南部」を選択。

もし「スペースの作成」という  
ウインドウが表示された場合は  
任意の名前（devなど）を指定  
してスペースを作成して下さい。



- 右上部の「カタログ」をクリックします。



# Node-RED環境の作成

- 左側の「アプリ」の下の「ボイラーブレット」を選択。
- 右側の方にある「Node-RED Starter」を選択します。

The screenshot shows the IBM Bluemix Catalog interface. On the left, there's a sidebar with categories like Compute, Storage, Network, Security, and App. Under the App category, 'ボイラーブレット' (Boilerplate) is selected and highlighted with a red box. In the main content area, there are several application starters listed in a grid. One of them, 'Node-RED Starter', is also highlighted with a red box. The other starters shown include ASP.NET Core Cloudant Starter, Internet of Things Platform Starter, IoT for Electronics Starter, Java Cloudant Web Starter, Java Workload Scheduler Web Starter, LoopBack Starter, MobileFirst Services Starter, Node.js Cloudant DB Web Starter, Personality Insights Java Web Starter, Personality Insights Node.js Web Starter, StrongLoop Arc, and Ruby Sinatra.

Category	Application Starter	Description
Compute	ASP.NET Core Cloudant Starter	Cloudant NoSQL DB サービスを ASP.NET Core アプリケーションで使用します。
Storage	Java Cloudant Web Starter	Cloudant NoSQL DB サービスを 'Liberty for Java™' ランタイムと一緒に使用します。
Network	Java Workload Scheduler Web Starter	このアプリケーションでは、IBM Cloud で 'Liberty for Java™' ランタイムを使用して、Workload Scheduler を構成できます。
Security	LoopBack Starter	これは、API を構成するために使用されるオープン・ソース LoopBack フレームワークに基づくサンプルです。
App	MobileFirst Services Starter	Bluemix のモバイル・サービスで、次のモバイル・アプリケーションの構築を始めてください。
	Node.js Cloudant DB Web Starter	Cloudant NoSQL DB サービスを 'SDK for Node.js™' ランタイムと一緒に使用します。
	Personality Insights Java Web Starter	A simple Java app that uses the Personality Insights service to analyze text to derive personality traits.
	Personality Insights Node.js Web Starter	A simple Node.js app that uses Personality Insights to analyze text to derive personality traits.
	StrongLoop Arc	このアプリケーションは、Node アプリのビルド、プロファイル作成、およびモニターを行うためのツールです。
	ボイラーブレット	今すぐ新しいアプリの作成を始めましょう。
	Node-RED Starter	This application demonstrates how to run the Node-RED open-source project within IBM Bluemix.
	Ruby Sinatra	Sinatra フレームワークを使用して Ruby Web アプリケーションを開発します。

# Node-RED環境の作成

- アプリケーション名を入力し、作成をクリックします。

すべて表示

## Cloud Foundry アプリの作成

Node-RED Starter

This application demonstrates how to run the Node-RED open-source project within IBM Bluemix.

アプリ名:  
固有の名前を入力してください

ホスト名: ドメイン名:  
固有の名前を入力してください mybluemix.net

コミュニティー

資料の表示

選択済みプラン:

SDK for Node.js™ Cloudant NoSQL DB

ヘルプが必要ですか?  
Bluemix IBM Firefox お問い合わせ

月額費用の計算  
費用計算

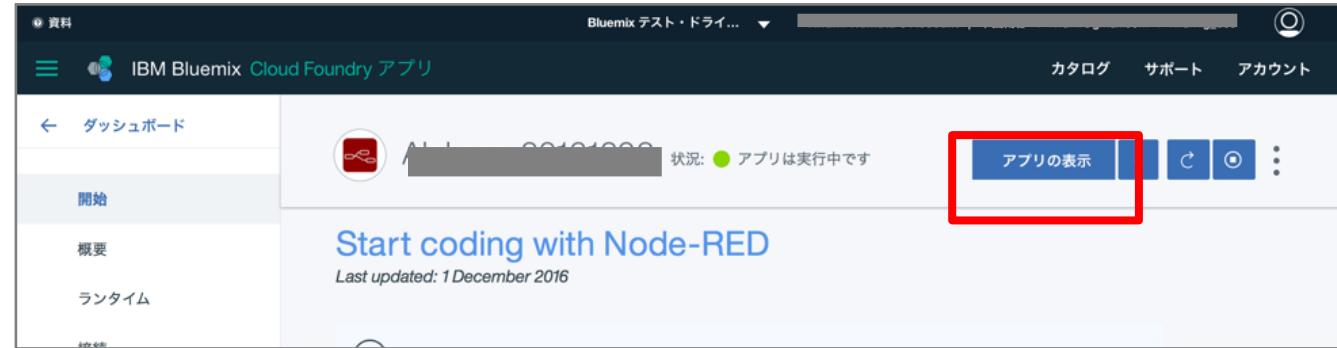
作成

# Node-RED環境の作成

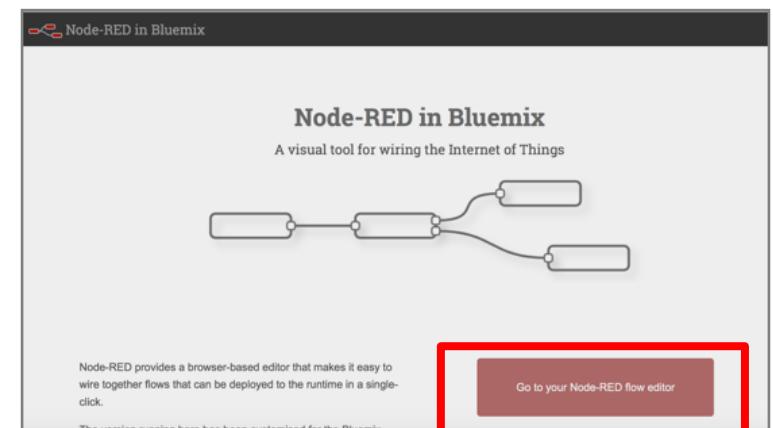
- ステージングが完了したら、作成した環境にアクセスしてみましょう。

(しばらく待ってもアプリが実行されない場合は、「アプリの表示」の右の「再始動」「始動する」などを実行してみて下さい。)

- 「アプリの表示」をクリックします。



- 「Go to your Node-RED flow editor」をクリックします。



- Node-REDが起動すればOKです。



# 本日のハンズオン

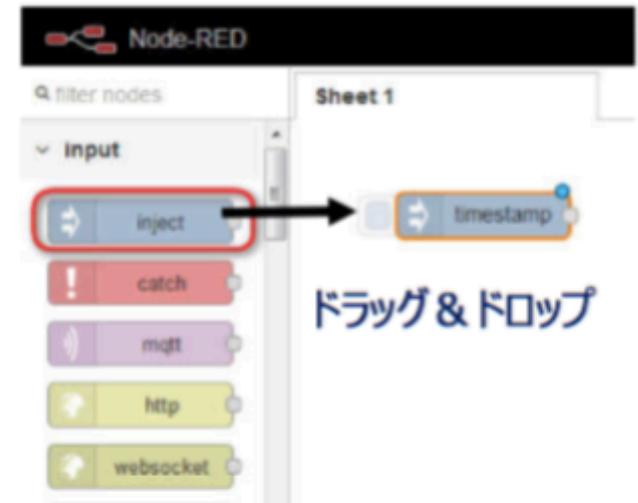
- Bluemixへのログイン
- Node-REDでHello World!
- IBM Watsonとは
- Watson Visual Recognition APIを使った画像認識アプリの作成

(\*) この章は、以下のURLにある資料のGUIを2016/12時点のものに修正したものです。  
[https://www.ibm.com/developerworks/community/wikis/home?lang=ja#!/wiki/Wde01e50fbfa\\_493c\\_8a88\\_6dd85c4d983f/page/%E5%AD%A6%E7%BF%92%E7%94%A8%E6%95%99%E6%9D%90](https://www.ibm.com/developerworks/community/wikis/home?lang=ja#!/wiki/Wde01e50fbfa_493c_8a88_6dd85c4d983f/page/%E5%AD%A6%E7%BF%92%E7%94%A8%E6%95%99%E6%9D%90)

# Node-REDでHello World – Step1

■ まずは処理を開始するノードを作つてみましょう。

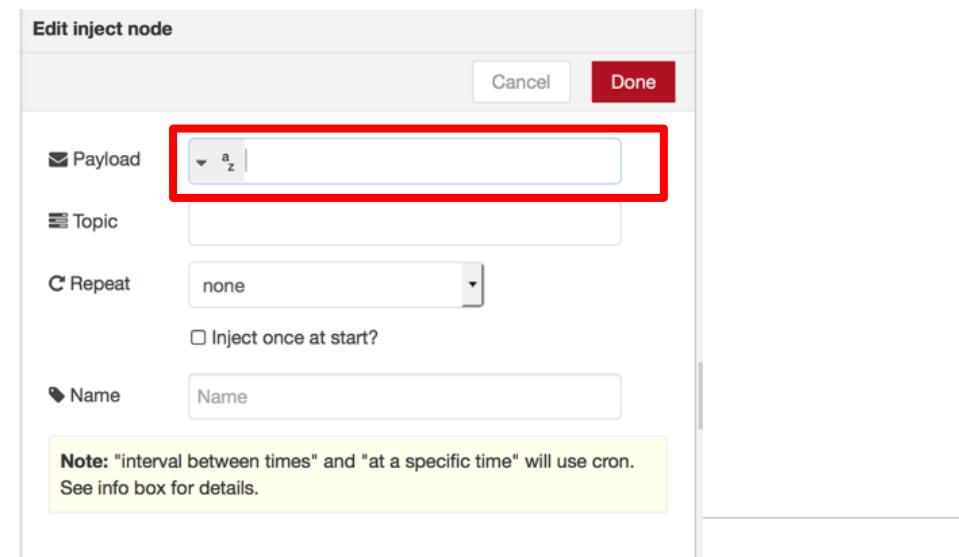
左側のパレットから「inject」ノードを中心のキャンバスに  
ドラッグ & ドロップします。



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

一番上のPayloadを「string」に設定します。  
イベントの起動を行うだけで次のノードには何も  
送信されません。

「Done」をクリックします。



# Node-REDでHello World – Step1

- 次にメッセージの表示内容を定義するノードを作つてみましょう。

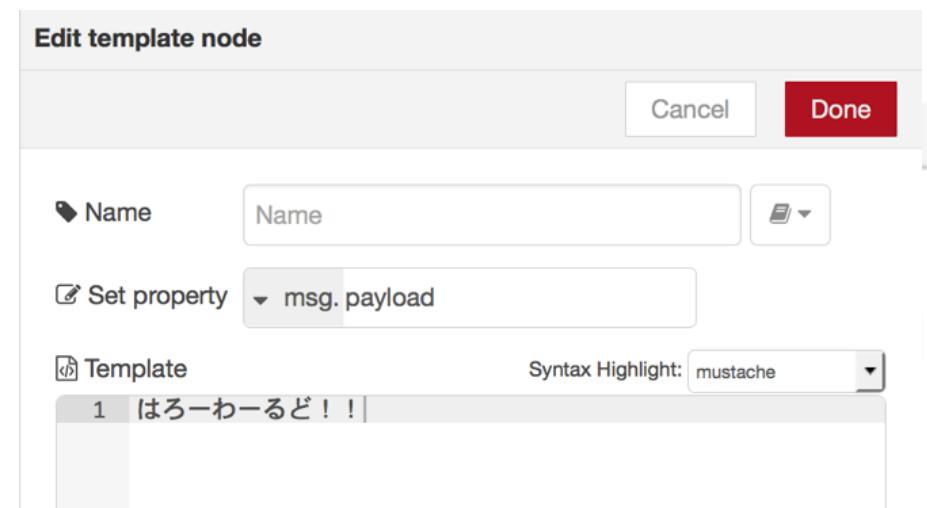
パレットから「template」ノードをキャンバスにドラッグ & ドロップします。



- 「template」ノードをダブルクリックして設定画面を開きます。

もともと入力されている文字列を削除し、「Hello World!!」など、任意の文字列を入力してください。

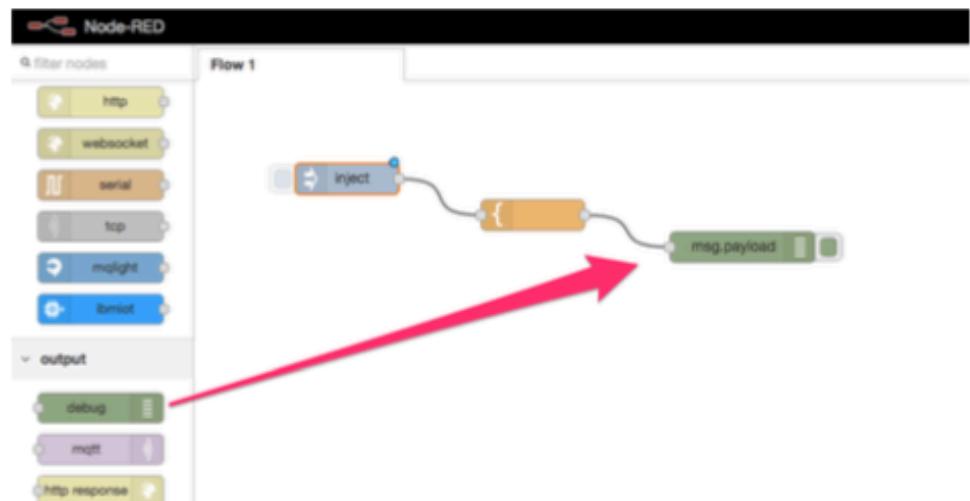
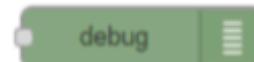
「Done」をクリックします。



# Node-REDでHello World – Step1

- 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、3つのノードを線で繋ぎます。

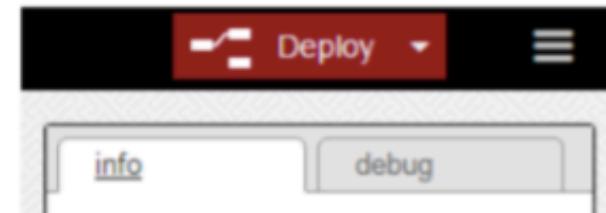


この部分を掴んで引っ張る

# Node-REDでHello World – Step1

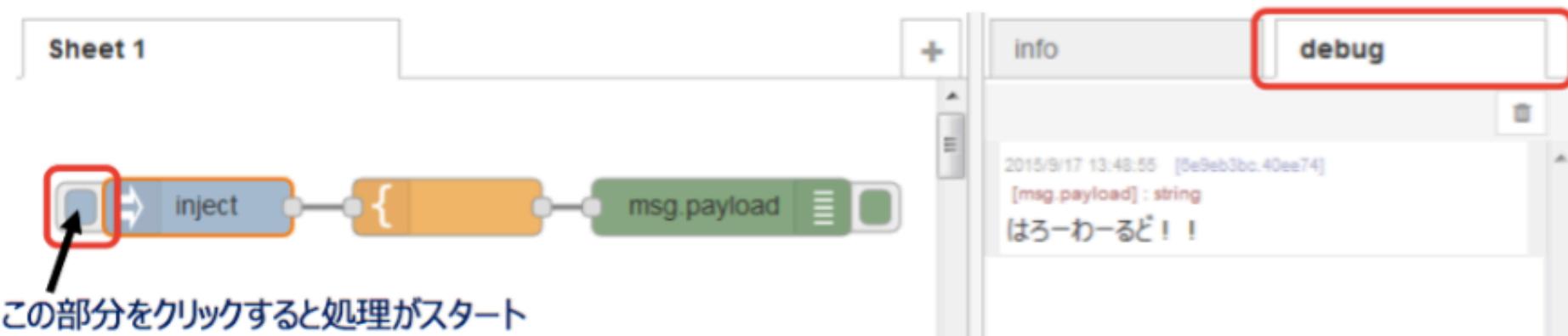
- これでHello Worldを表示するフローが完成しました。

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。



- 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のコンソールに表示されます。  
injectノードの左側にあるボタンをクリックすると処理が起動します。  
Hello World!! の文字列がdebugコンソールに表示されれば成功です。



## Node-REDでHello World – Step2

- 作成したフローを拡張してみましょう。  
Hello Worldの文字列をWebページに出力してみます。

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



- 「http」ノードをダブルクリックして設定画面を開きます。

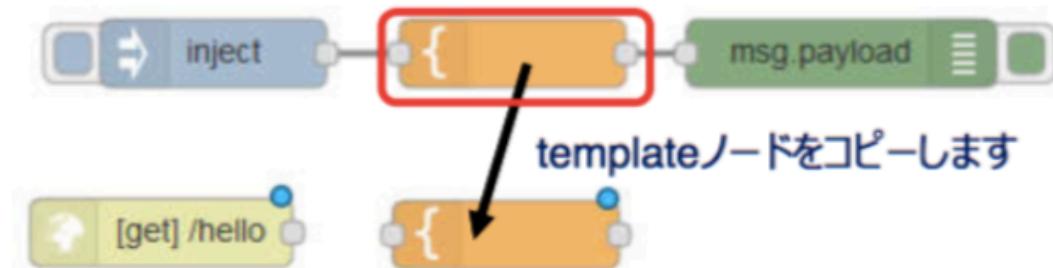
URLとして「/hello」と入力します。  
このパスにWebブラウザでアクセスし、Hello Worldの文字列を表示させます。

Edit http in node

# Node-REDでHello World – Step2

- 「template」ノードは先ほど作成したノードをコピーします。

ノードをクリックして選択し、Ctrl + C でコピーできます。



- パレットから「http response」ノードをキャンバスにドラッグ & ドロップします。



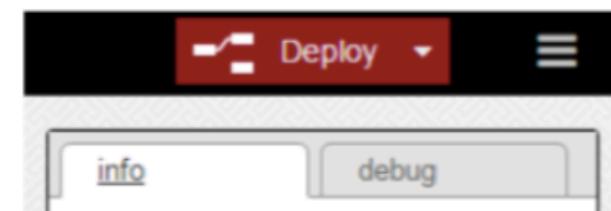
## Node-REDでHello World – Step2

- 3つのノードを線で繋ぎます。



- これでHello WorldをWebに表示するフローが完成しました。

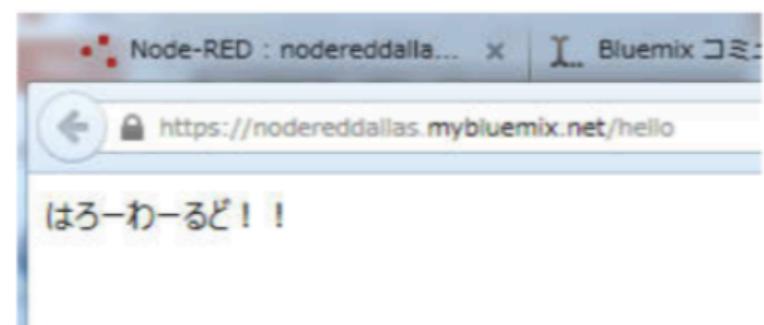
それでは実際に動かしてみましょう。  
右上の「Deploy」ボタンをクリックします。



- Webブラウザの新しいタブを開きます。  
Node-REDフローエディタのURLの  
末尾の「red/#」を削除し、代わりに  
「hello」と入力してWebページに  
アクセスします。

(例)

<https://node-red.xxx/red/#> の場合は、  
<https://node-red.xxx/hello> を開きます。



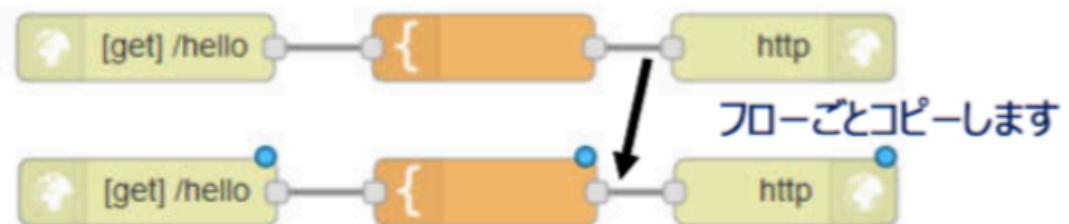
Hello WorldがWebブラウザに表示されました！

# Node-REDでHello World – Step 2

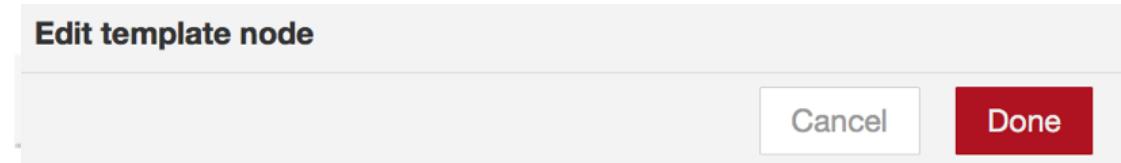
- あらかじめtemplateノードに書かれた文字列だけでなく、任意の文字列を表示できるようにしてみましょう。そのためには変数を使用します。  
まずはURLリクエストパラメータで、文字列を変数として渡す方法を試してみます。

- Step2で作成したフローをコピーします。

Shiftキーを押しながらノードをクリックすると  
フロー全体が選択されます。



- 左端のhttpノードの設定画面を開き、URLとして「/hello2」と入力します。



- 中央のtemplateノードの設定画面を開き、表示内容を修正します。

{{name}}さん、はろーわーるど！！

と変更しました。

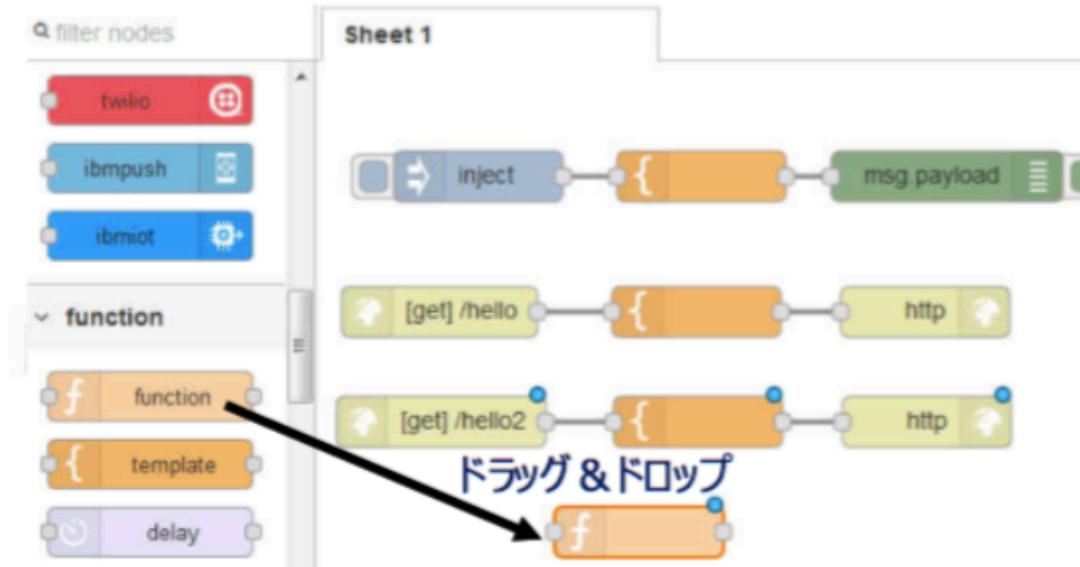
Name	Name	<input type="button" value="edit"/>
Set property	msg.payload	
Template	Syntax Highlight: mustache	
1 {{name}}さん、はろーわーるど！！		

# Node-REDでHello World – Step2

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



functionノードには JavaScriptでコードを書くことができます。



- 「function」ノードをダブルクリックして設定画面を開きます。

一行目に  
msg.name = msg.payload.name;  
と入力します。

二行目はそのままDoneをクリックします。

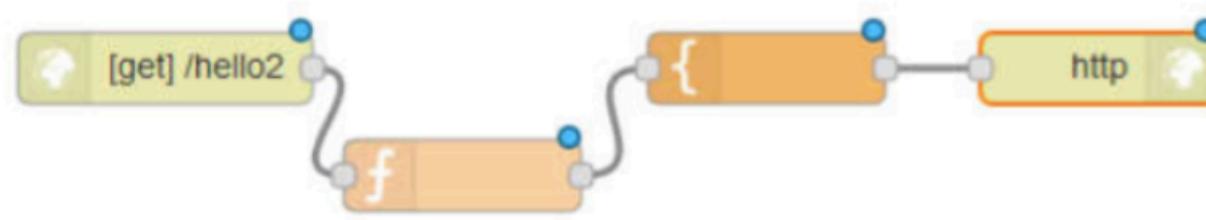
## Function

```
1 msg.name = msg.payload.name;
2 return msg;
```

## Node-REDでHello World – Step2

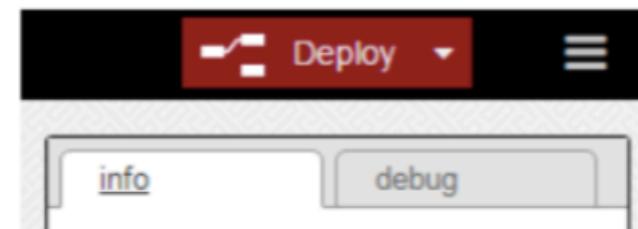
■ ノードを下の図のようにつなぎ直します。

線を削除するには、クリックで線を選択してDELETEキーです。



■ これでURLのパラメータで変数の値を受け取って、Webに文字列を表示する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Hello World – Step2

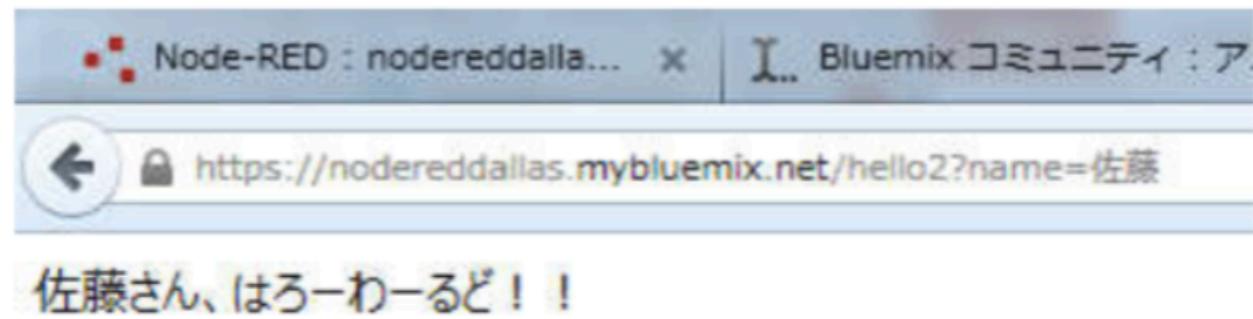
- Webブラウザの新しいタブを開いて、以下のURLにアクセスしてみましょう。

https://node-red.xxx/hello2?name=佐藤

Node-REDエディタのURLです

name変数の値として「佐藤」という文字列を渡しています

- これでURLのパラメータで変数の値を受け取って、Webに文字列を表示するフローが完成しました。

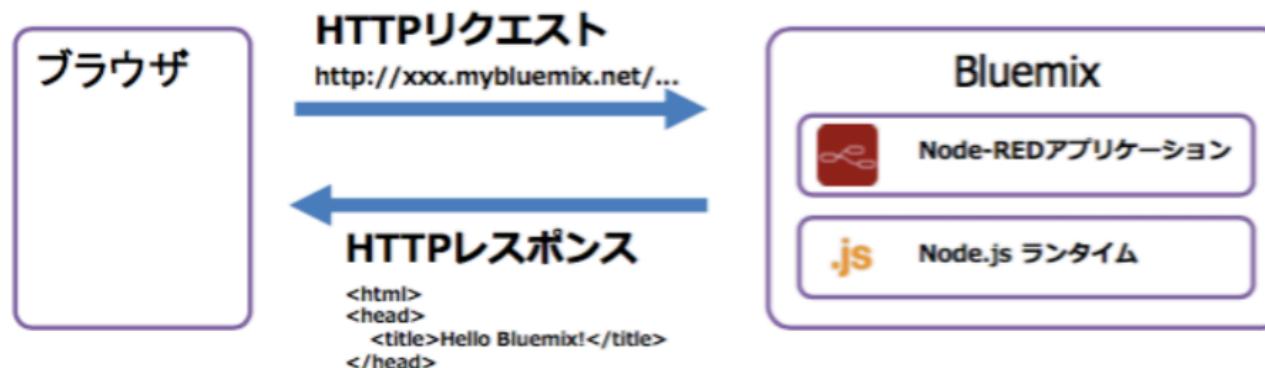


パラメータで渡した名前がWebブラウザに表示されました！

## Node-REDでHello World – Step3

- ここではNode-REDを使用して、Webアプリケーションを作成していきます。

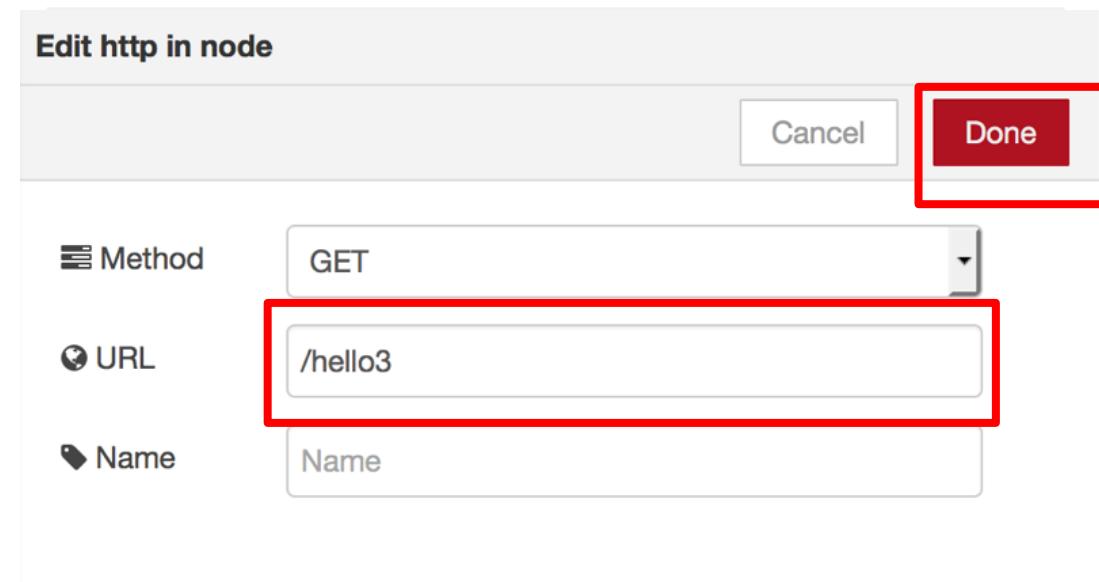
- HTMLフォームの出力
  - ✓ ブラウザからのHTTPリクエストに対してHTTPレスポンスを返す方法を学びます。
- HTMLフォームで入力された値の取得
  - ✓ HTTPリクエストのパラメータを取得して、HTTPレスポンスに出力する方法を学びます。



## Node-REDでHello World – Step3

- HTTPリクエストを受ける HTTP in ノードを定義します。
- 左側のリソースパレットの inputカテゴリ内のhttpノードをフローエディタ中央のキャンバスにドラッグ&ドロップし、ダブルクリックします。
- 各属性を修正して、「Done」ボタンをクリックします。

Method: GET  
URL: /hello3



## Node-REDでHello World – Step3

- HTMLを定義するtemplateノードを定義します。
- 左側のリソースパレットの functionカテゴリ内のtemplateノードをフロー エディタ中央のキャンバスにドラッグ&ドロップしダブルクリックします。

- 各属性を修正します。

Name: 入力画面

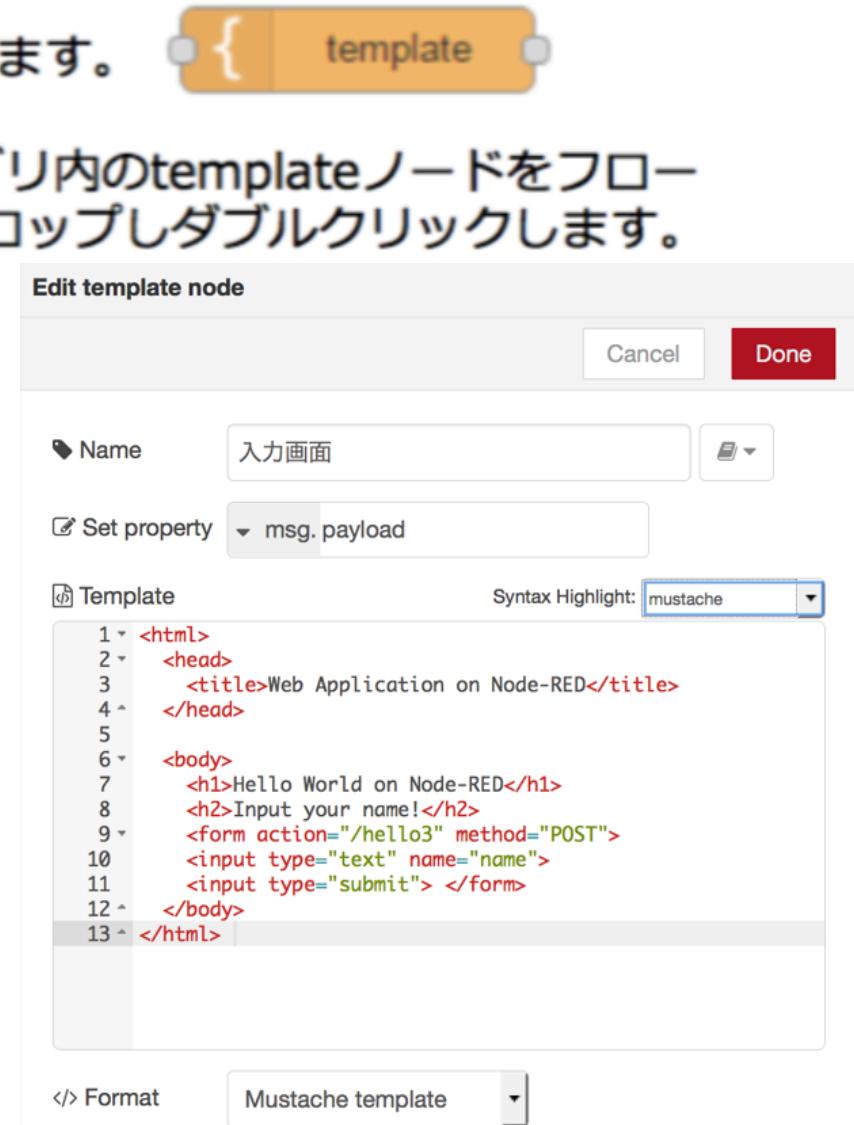
Set property: msg.payload

Syntax Highlight: HTML

Template: HTMLを記載

Format: Mustache template

- 「Done」ボタンをクリックします。
- HTTP inノードと線を繋ぎます。



# Node-REDでHello World – Step3

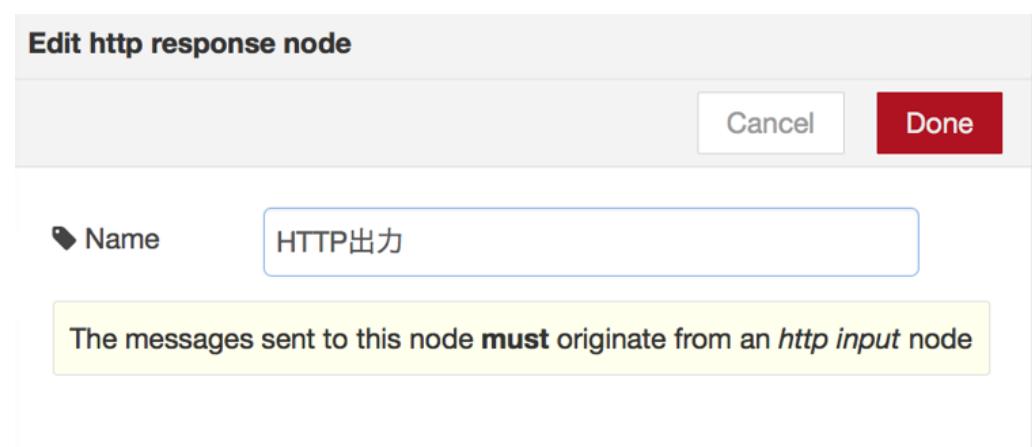
## ■ Templateノードの記述例：入力画面

```
<html>
  <head>
    <title>Web Application on Node-RED</title>
  </head>
  <body>
    <h1>Hello World on Node-RED</h1>
    <h2>Input your name!</h2>
    <form action="/hello3" method="POST">
      <input type="text" name="name">
      <input type="submit"> </form>
    </body>
</html>
```

# Node-REDでHello World – Step3

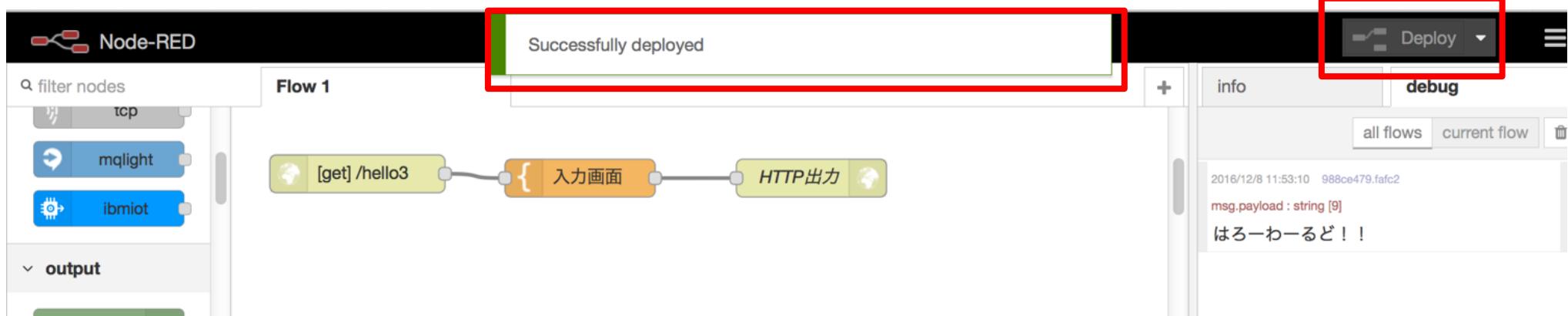
- HTTPレスポンスとなるHTTP responseノードを定義します。  

- 左側のリソースパレットの outputカテゴリ内のhttpノードをフローエディタ中央のキャンバスにドラッグ&ドロップし、ダブルクリックします。
- 各属性を修正します。  
Name: HTTP出力
- 「Done」ボタンをクリックします。
- 入力画面と線を繋ぎます。

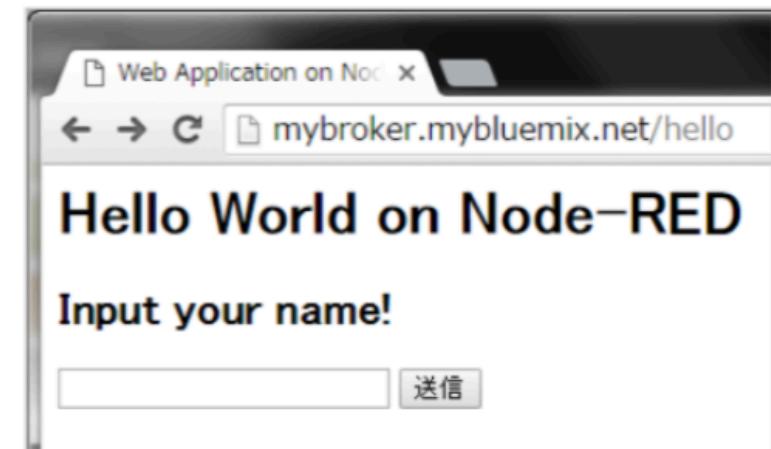


## Node-REDでHello World – Step3

- これまでのステップで下図のようなフローができあがります。右上の「Deploy」ボタンをクリックし、アプリケーションをデプロイします。上部に“Successfully deployed”と表示されれば、Bluemix 上でのアプリケーションのデプロイは成功です。

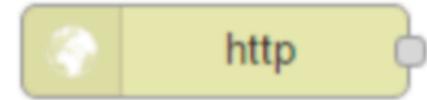


- ブラウザから確認  
Node-REDのボイラーテンプレートの名前と  
http in ノードで指定したURLをもとにアクセス。  
ブラウザからアクセス。  
例) <http://XXX.mybluemix.net/webtest>



# Node-REDでHello World – Step3

- HTTPリクエストを受ける HTTP in ノードを定義します。



- 左側のリソースパレットの inputカテゴリ内のhttpノードをフローエディタ中央のキャンバスにドラッグ&ドロップし、ダブルクリックします。
- 各属性を修正して、「Done」ボタンをクリックします。

Method: POST

URL : /hello3

Edit http in node

Cancel

Done

Method

POST

URL

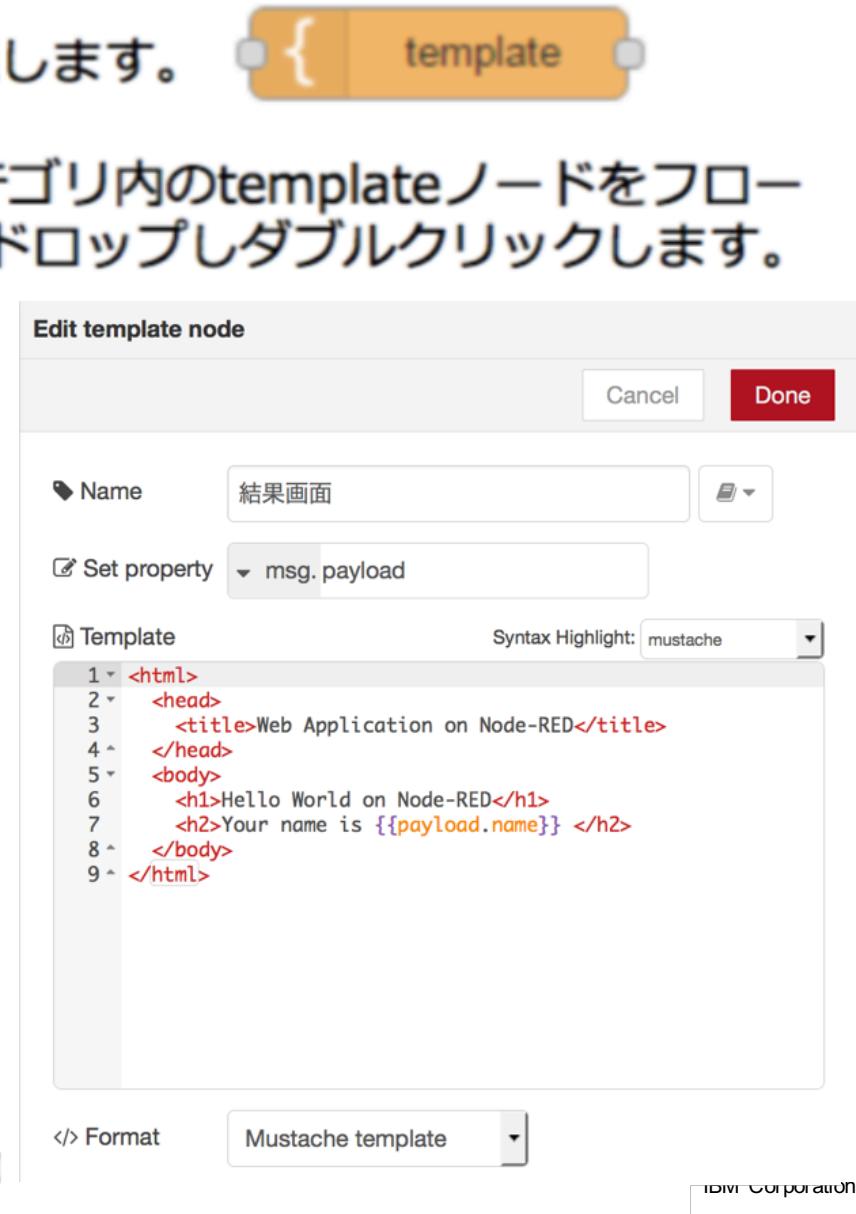
/hello3

Name

Name

# Node-REDでHello World – Step3

- HTMLを定義するtemplateノードを定義します。
- 左側のリソースパレットの functionカテゴリ内/templateノードをフローエディタ中央のキャンバスにドラッグ&ドロップしダブルクリックします。
- 各属性を修正します。
  - Name: 結果画面
  - Set property: msg.payload
  - Syntax Highlight: HTML
  - Template: HTMLを記載
  - Format: Mustache template
- 「Done」ボタンをクリックします。
- HTTP inノードと線を繋ぎます。



# Node-REDでHello World – Step3

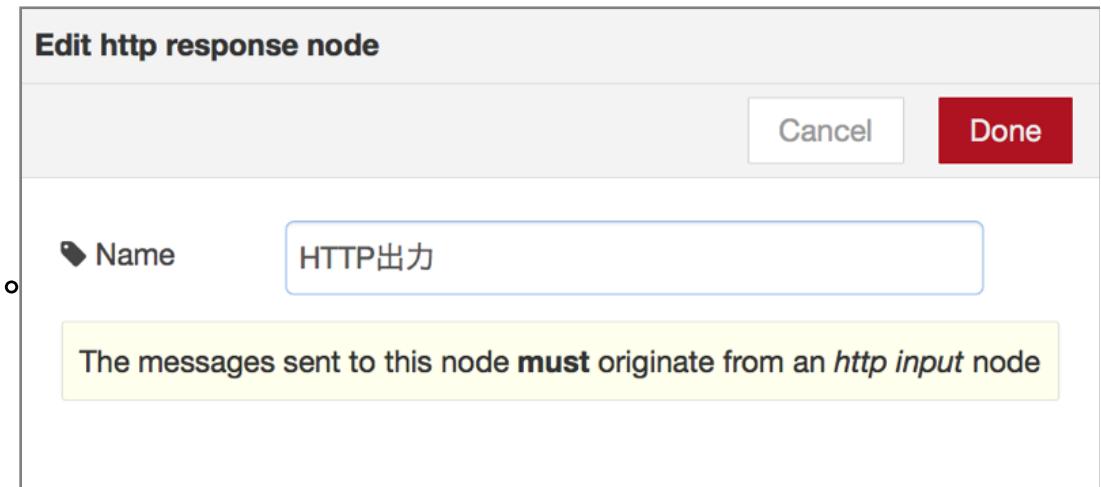
- Templateノードの記述例：出力結果

```
<html>
  <head>
    <title>Web Application on Node-RED</title>
  </head>
  <body>
    <h1>Hello World on Node-RED</h1>
    <h2>Your name is {{payload.name}} </h2>
  </body>
</html>
```

## Node-REDでHello World – Step3

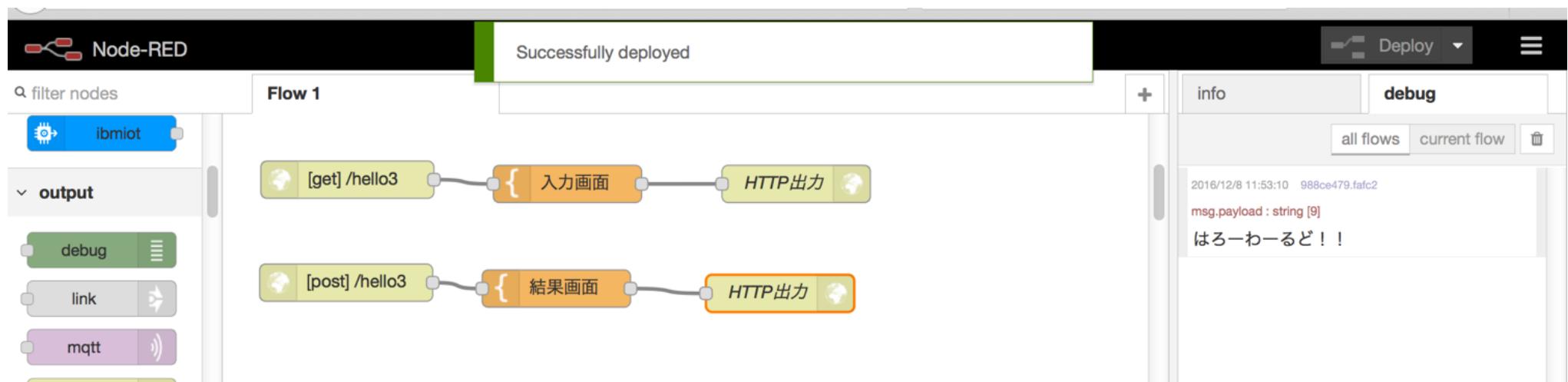
- HTTPレスポンスとなるHTTP responseノードを定義します。  

- 左側のリソースパレットの outputカテゴリ内のhttpノードをフローエディタ中央のキャンバスにドラッグ&ドロップし、ダブルクリックします。
- 各属性を修正します。  
Name: HTTP出力
- 「Done」ボタンをクリックします。
- 入力画面と線を繋ぎます。



## Node-REDでHello World – Step3

- これまでのステップで下図のようなフローができるようになります。右上の「Deploy」ボタンをクリックし、アプリケーションをデプロイします。上部に“Successfully deployed”と表示されれば、Bluemix 上でのアプリケーションのデプロイは成功です。

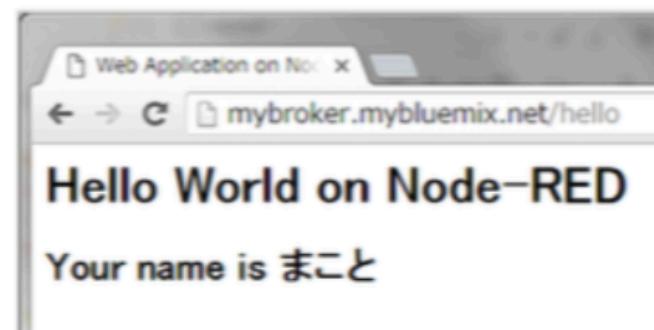


## Node-REDでHello World – Step3

### ■ HTMLフォームにデータを入力

最初に作成したHTMLフォームを出力するURLへアクセスして、  
データを入力して送信ボタンをクリック。

<http://xxxx.mybluemix.net/hello3>



### ■ /hello3にHTTPリクエストが送信され、 以下の画面が表示されます。

# 本日のハンズオン

- Bluemixへのログイン
- Node-REDでHello World!
- IBM Watsonとは
- Watson Visual Recognition APIを使った画像認識アプリの作成