



# Node-REDのノード開発ハンズオン

2019/04/22

日立製作所 中央研究所

横井 一仁

ハンズオン開始前に以下のソフトウェアをインストールしておいてください。

- Node.js v10.x LTS (<https://nodejs.org/ja/download>)
- Node-RED v0.20.x  
コマンドプロンプト上で「npm install -g node-red」を実行
- Node generator v0.0.4  
コマンドプロンプト上で「npm install -g node-red-nodegen」を実行
- Windows-Build-Tools (Windowsの場合)  
管理者モードでコマンドプロンプトを起動し、  
「npm install -g --production windows-build-tools」を実行
- Xcode (macOSの場合)
- node-red-contrib-web-worldmap、node-red-contrib-browser-utilsノード  
Node-REDフローエディタのメニュー ->「パレットの管理」からインストール
- テキストエディタ(Visual Studio Codeなど)

下記リポジトリにあるハンズオン説明スライドも開いておいてください。

<https://github.com/kazuhiyokoi/nodegen-handson>

## コンテンツ

---

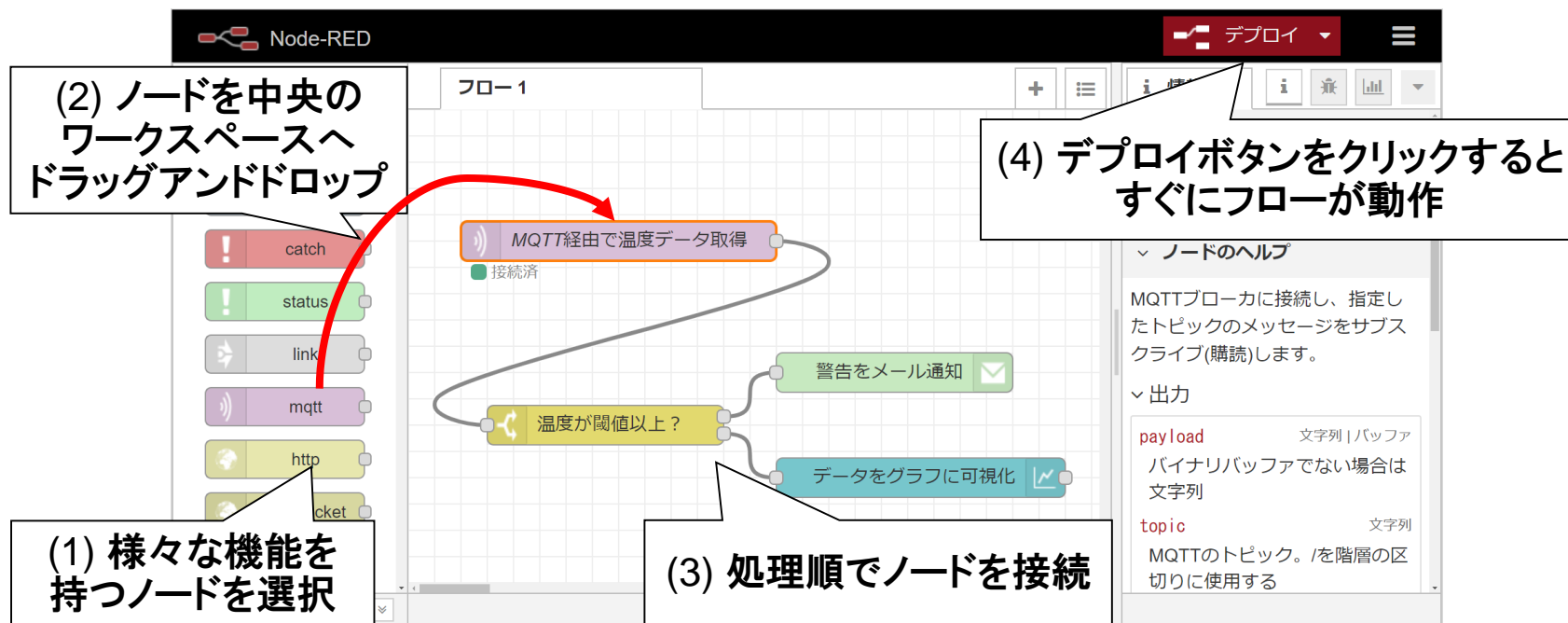
- Node generatorの説明
- ハンズオン1: Open APIドキュメントからオリジナルノードを作成
- ハンズオン2: functionノードからオリジナルノードを作成



## Node generatorの説明

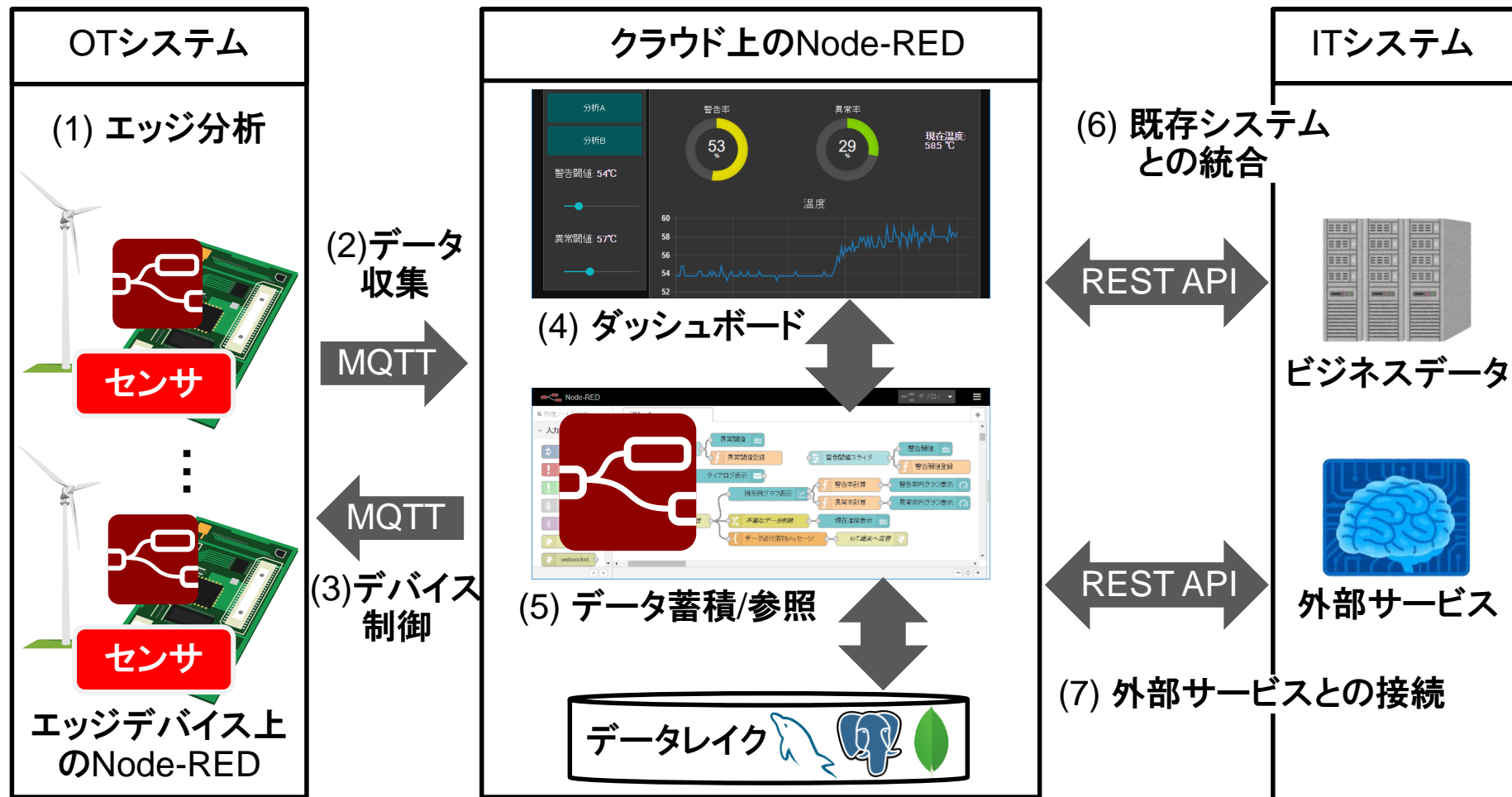
## IoTアプリケーション向けのビジュアルプログラミングツール

- 迅速な開発を可能とするフローベースドプログラミング環境
- 新機能を追加できる様々なノード(部品)が存在
- Linux Foundation(JS Foundation)管理下のオープンソースソフトウェア



## Node-REDフローエディタ

様々な外部システムやデバイス、ライブラリとつなぐノードを活用することで素早くアプリケーションを開発できる。



- 自社/他社サービスとの接続のため、オリジナルノードの開発が必要となることもある
- 1個のノードは数千行のコード量であるため、ノード開発時間大

数千行のコード

ノード開発時間大

JavaScript  
ファイル

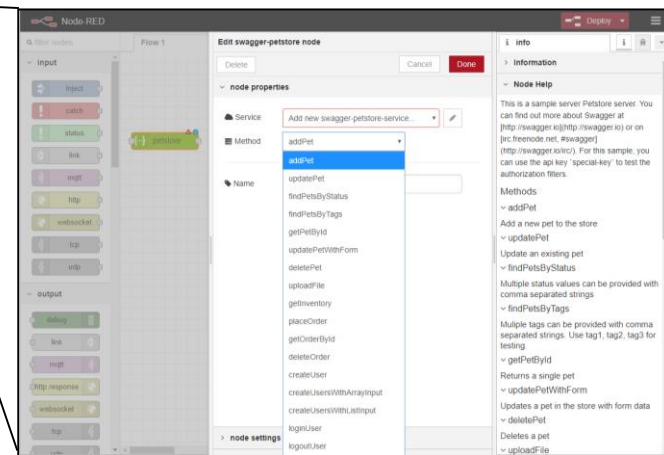
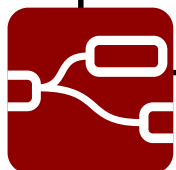
HTML  
ファイル



オリジナルノード

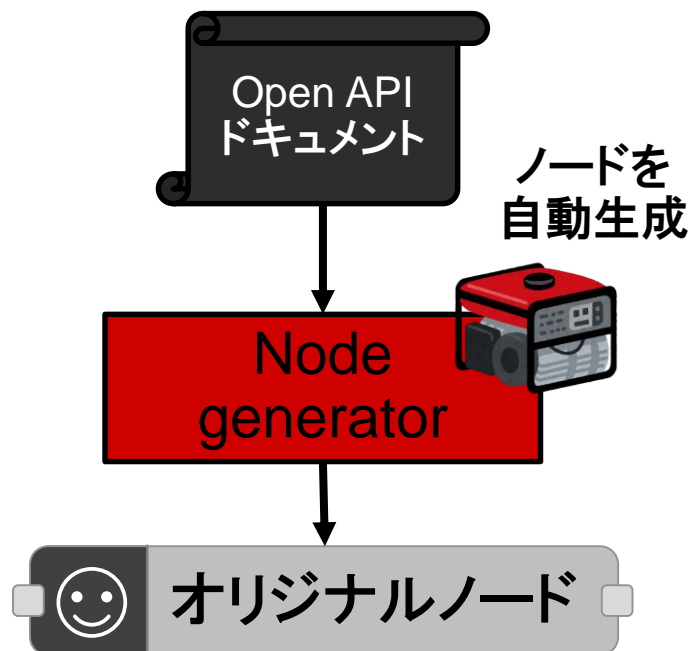
Node.js上  
で実行

Node-RED  
ランタイム

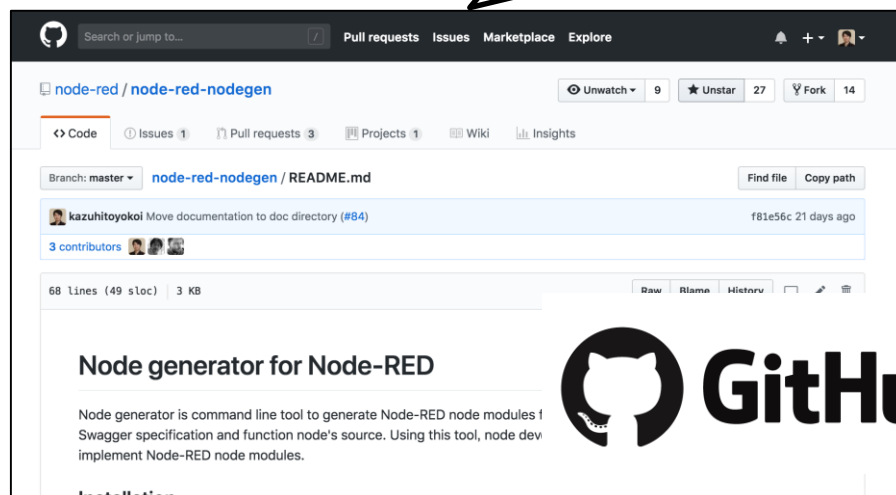


Node-REDフローエディタ  
のUI

- Node generatorは、Open APIドキュメントやfunctionノードのソースコードから、ノードを自動生成できるツール
- Linux Foundation (JS Foundation) のオープンソースソフトウェア



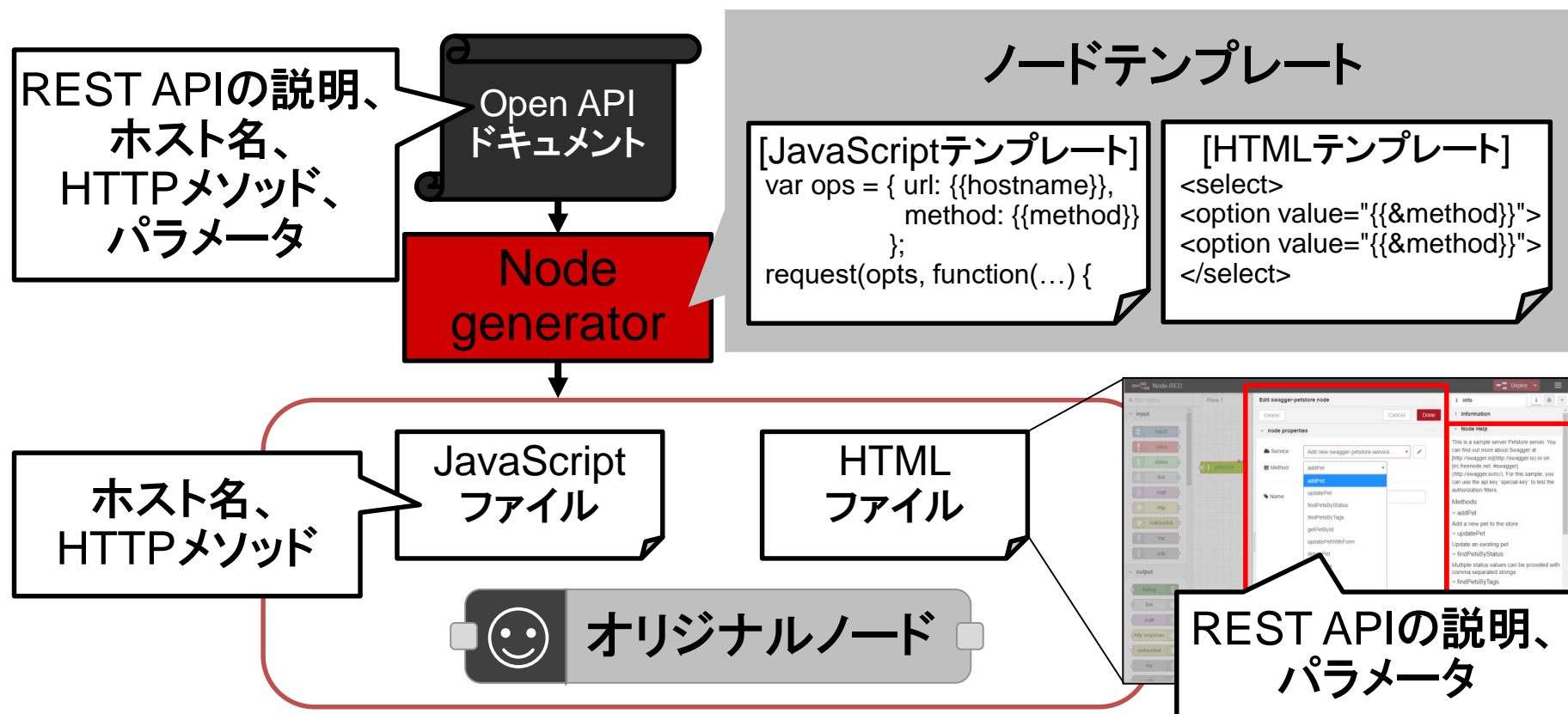
“Node-RED Node generator”  
で検索すると見つかります



<https://github.com/node-red/node-red-nodegen>



- Node generatorは、内部にてREST API固有の情報を除いた共通のクライアントコードを持つノードテンプレートを保持
- Node generatorは、本ノードテンプレートにREST API固有の情報を挿入することでノードを生成



Node generatorは、以下のディレクトリ、ファイルを出力




node-red-contrib-<ノード名>

- ├ node.js (サーバ側のNode.jsで動作するプログラム)
- ├ node.html (ノードの見た目、ノードプロパティUI、情報タブを記述)
- ├ locales
  - │ └ en-US — node.json (ノードプロパティUIの英語メッセージ)
  - │ └ ja — node.json (ノードプロパティUIの日本語メッセージ)
  - │ └ zh-CN — node.json (ノードプロパティUIの中国語メッセージ)
  - │ └ de-DE — node.json (ノードプロパティUIのドイツ語メッセージ)
- ├ icons
  - │ └ icon.png (ノードのアイコンファイル)
- ├ test
  - │ └ node\_spec.js (ノードのテストケース)
- ├ LICENSE (ライセンスファイル)
- ├ package.json (ノードのパッケージ情報)
- └ README.md (ノードの使い方等のドキュメント)

## IBMが画像認識、音声認識、自然言語処理を行うDockerコンテナにアクセスするノードをNode generatorを用いて開発

**IBM Developer** Topics ▼ Community ▼ More open source at IBM ▼

 IoT

**CODE**  
Models  
Code Patterns  
Open Projects  
  
**CONTENT**  
Announcements  
Articles  
Series  
Tutorials  
Videos

**TUTORIAL**  
  

# Use Node-RED Node Generator to create new nodes from APIs and services

Access a RESTful deep learning microservice from your Node-RED flows

Nick Kasten | Published March 25, 2019

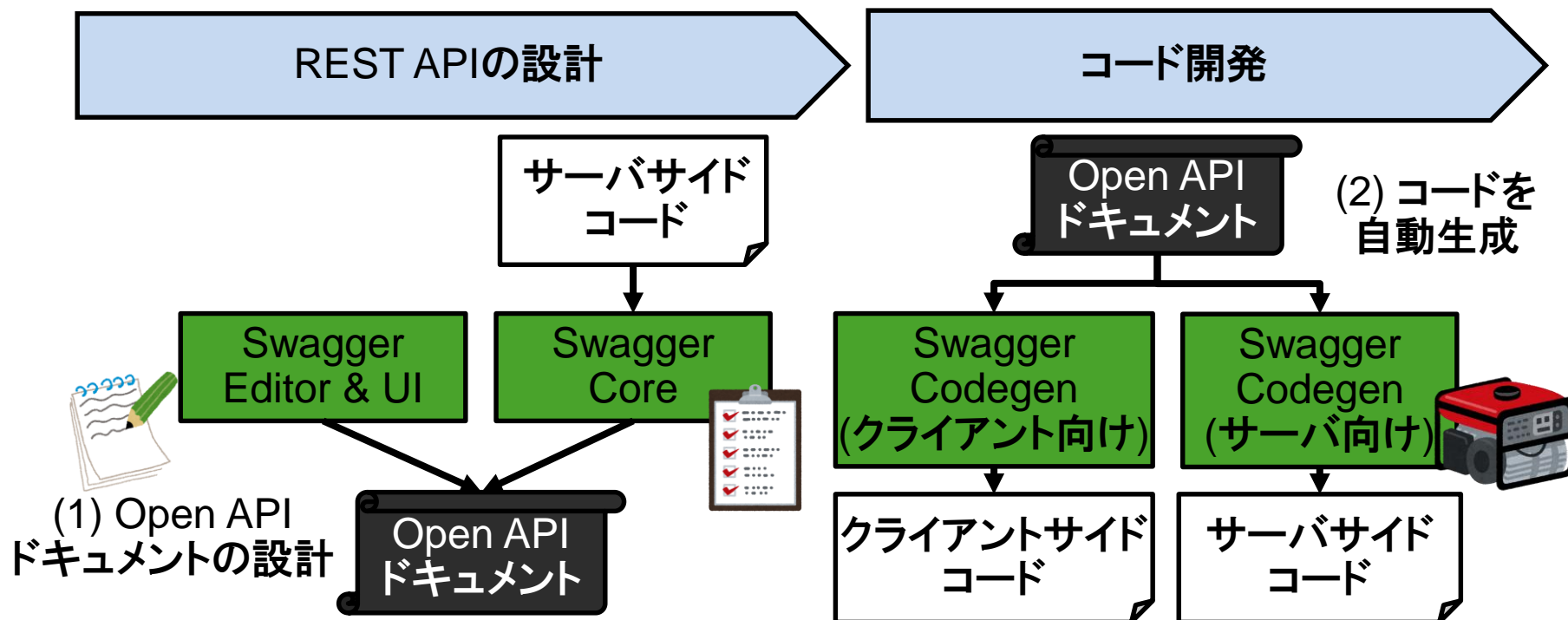
## Node generatorの紹介ページ

<https://developer.ibm.com/tutorials/use-node-red-node-generator-to-create-new-nodes-from-apis-and-services/>



## ハンズオン1: Open APIドキュメントから オリジナルノードを作成

- REST API定義を記述する業界標準のフォーマット
- Open APIプロジェクトにより提供されているツール
  - Swagger Editor & UI: Open APIドキュメントを記述するためのエディタ、REST APIドキュメントの自動生成
  - Swagger Core: サーバサイドのコードからOpen APIドキュメントを自動生成
  - Swagger Codegen: Open APIドキュメントからコードを自動生成



Open APIドキュメントには、REST APIのアクセス方法  
についての情報が記載されている。

REST API  
の説明

```
swagger: '2.0'
info:
  description: 'This is a sample server Petstore server.'
  version: 1.0.0
  title: Swagger Petstore
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
host: petstore.swagger.io
basePath: /v2
paths:
  /store/inventory:
    get:
      summary: Returns pet inventories by status
      description: Returns a map of status codes to quantities
      operationId: getInventory
      produces:
        - application/json
      parameters: []
      responses:
        '200':
          description: successful operation
```

ホスト名

HTTP  
メソッド

パラメータ

## 宇宙ステーションを位置情報を取得するノードを作成

- (1) 位置情報取得APIのOpen APIドキュメントからノードを自動生成
- (2) 世界地図上に宇宙ステーションの位置を表示するフローを作成

宇宙ステーション



REST API

ウェブブラウザ



2. 緯度経度  
を取得

3. 位置情報データの  
フォーマット変換

4. 世界地図上に  
ピンを配置

Open API  
ドキュメント

1. ノード生成、  
インストール

Open APIドキュメント  
からノードを自動生成

- (1-1) ハンズオンのリポジトリ(<https://github.com/kazuhitoyokoi/nodegen-handson>)からOpen APIドキュメント(swagger.yaml)をホームディレクトリ(C:¥Users¥<ユーザ名>¥)へダウンロード

```
swagger: '2.0'
info:
  description: The International Space Station location
  version: 4.5.1
  title: ISS Location
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
host: api.open-notify.org
basePath: /
schemes:
  - http
paths:
  /iss-now.json:
    get:
      summary: Current ISS location over Earth (latitude/longitude)
      description: This is a simple API to return the current location of the ISS.
      operationId: ISSLocationNow
      produces:
        - application/json
      responses:
        '200':
          description: successful operation
```

ダウンロードしたOpen APIドキュメントの内容



(1-2) コマンドプロンプトを開き、ホームディレクトリに移動

```
> cd C:\Users\<ユーザ名>
```

(1-3) node-red-nodegenコマンドを実行しノードを生成

```
> node-red-nodegen swagger.yaml
```

(1-4) 生成したノードのディレクトリにディレクトリを変更

```
> cd node-red-contrib-iss-location
```

```
コマンドプロンプト
C:\Users\yokoi>node-red-nodegen swagger.yaml
Success: node-red-contrib-iss-location

C:\Users\yokoi>cd node-red-contrib-iss-location

C:\Users\yokoi\node-red-contrib-iss-location>dir
2019/04/18 12:10 <DIR> icons
2019/04/18 12:10 4,414 lib.js
2019/04/18 12:10 60 LICENSE
2019/04/18 12:10 <DIR> locales
2019/04/18 12:10 2,805 node.html
2019/04/18 12:10 2,530 node.js
```

ノードを構成するファイル  
が生成されている

## (1-5) シンボリックリンク作成の準備

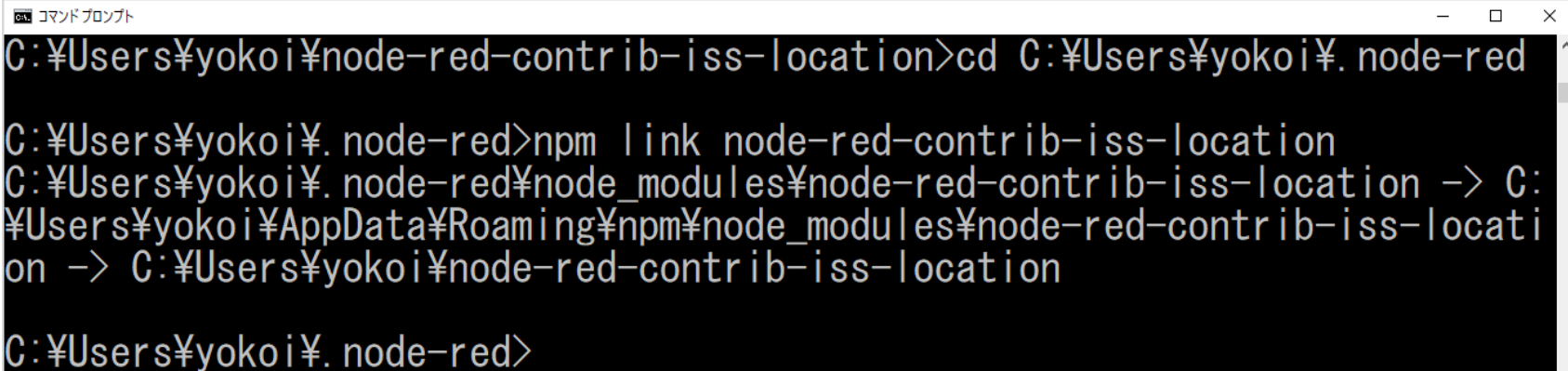
```
> npm link
```

## (1-6) Node-REDのホームディレクトリに移動

```
> cd C:¥Users¥<ユーザ名>¥.node-red
```

## (1-7) シンボリックリンクを作成

```
> npm link node-red-contrib-iss-location
```



```
コマンドプロンプト
C:¥Users¥yokoi¥node-red-contrib-iss-location>cd C:¥Users¥yokoi¥.node-red
C:¥Users¥yokoi¥.node-red>npm link node-red-contrib-iss-location
C:¥Users¥yokoi¥.node-red¥node_modules¥node-red-contrib-iss-location -> C:¥Users¥yokoi¥AppData¥Roaming¥npm¥node_modules¥node-red-contrib-iss-location -> C:¥Users¥yokoi¥node-red-contrib-iss-location
C:¥Users¥yokoi¥.node-red>
```

(2-1) Node-REDを起動(起動中の場合は再起動)

> node-red

(2-2) フローエディタ(<http://localhost:1880>)にアクセス(またはリロード)

-> パレットの「機能」カテゴリの中にiss locationノードが登場

(2-3) 生成されたノードをワークスペースにドラッグアンドドロップし、ノードプロパティUI上でメソッドを選択

The image shows the Node-RED web interface. On the left, the 'Node Palette' is visible with a search bar and a list of nodes. The 'iss location' node is highlighted with a red circle. A red arrow points from this node to the 'iss-location' node in the 'Workspace'. On the right, the 'iss-location ノードを編集' (Edit iss-location node) panel is open. It shows the 'メソッド' (Method) dropdown menu with 'ISSLocationNow' selected. A callout box with the text 'メソッドを選択' (Select method) points to the dropdown menu.

(2-4) injectノード、生成したノード、debugノードを順に接続したフローを作成し、デプロイボタンをクリック

(2-5) injectノードの右にあるボタンをクリックして、フローを実行  
→ デバッグタブに宇宙ステーションの現在の緯度(latitude)、経度(longitude)が出力される

The screenshot displays the Hitachi Flow Designer interface. On the left, a workflow canvas shows three nodes connected in sequence: a blue 'タイムスタンプ' (Timestamp) node, a green 'iss-location' node, and a green 'msg.payload' node. The 'msg.payload' node is highlighted with a red dashed border. On the right, the 'デバッグ' (Debug) tab is active, showing a log entry for the flow execution. The log entry includes a timestamp, a node ID, and a message object containing the ISS position data.

フロー1

タイムスタンプ

iss-location

msg.payload

デバッグ

2019/4/17 11:50:00 node: bfdb38f7.544468

msg.payload : Object

- object
  - timestamp: 1555469300
  - message: "success"
  - iss\_position: object
    - latitude: "-45.9101"
    - longitude: "2.8959"

REST APIから取得した位置情報が表示される

## (2-6) injectノードのプロパティUIで以下の繰り返し処理を設定

#	項目	設定値
1	繰り返し	指定した時間間隔
2	時間間隔	1秒

Node-RED

フロー 1

inject ノードを編集

削除 中止 完了

プロパティ

ペイロード

トピック

Node-RED起動 0.1 秒後、以下を行う

繰り返し 指定した時間間隔

時間間隔 1 秒

名前

1秒間隔でフローを実行するよう設定

## (2-7) 生成したノードの後ろにchange、world mapノード(左側に端子)を接続

## (2-8) changeノードに変数の代入を設定し、デプロイボタンをクリック

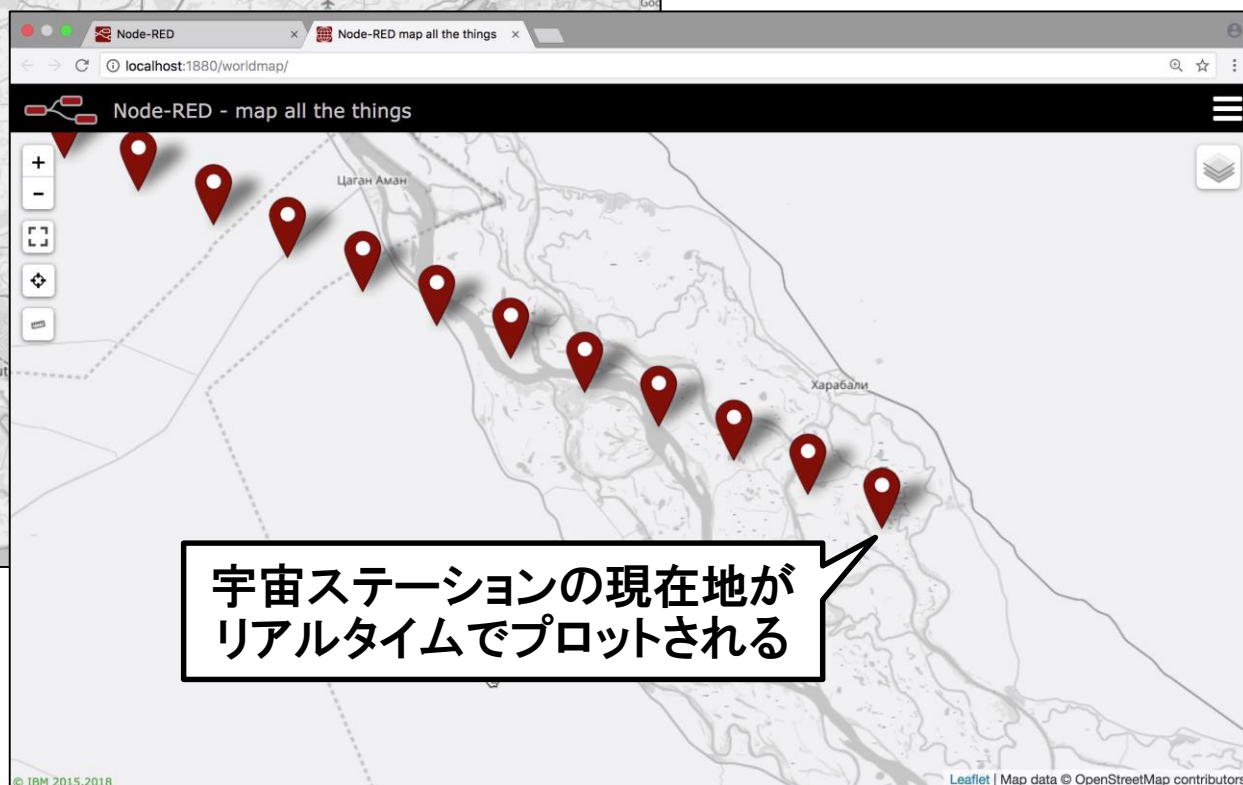
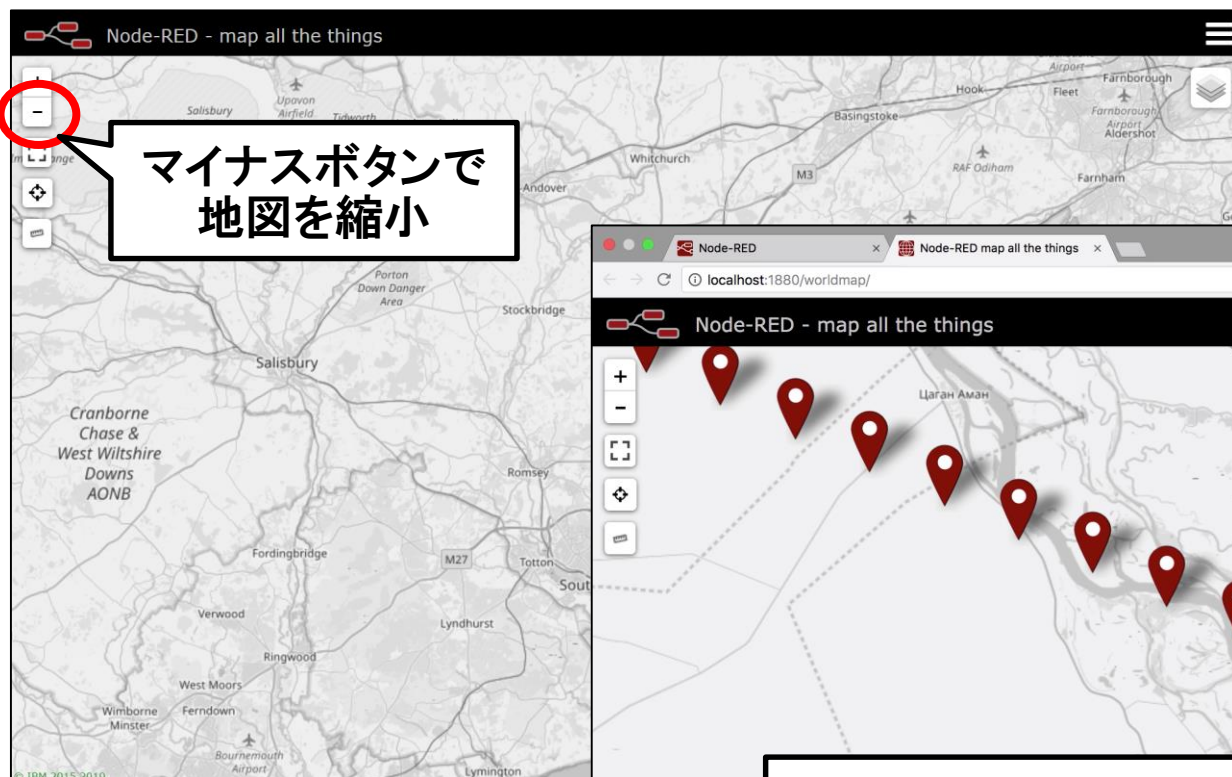
#	プルダウンメニュー	1つ目の設定値	対象の値
1	値の代入	msg.payload.lat	msg.payload.iss_position.latitude
2	値の代入	msg.payload.lon	msg.payload.iss_position.longitude
3	値の代入	msg.payload.name	msg.payload.timestamp

The screenshot shows the Node-RED interface. On the left, the 'location' category is expanded, showing 'worldmap' and 'tracks' nodes. In the center, a flow is visible with a 'タイムスタンプ' (Timestamp) node, an 'iss-location' node, and a 'set msg.payload' node. The 'set msg.payload' node is connected to a 'worldmap' node. A callout points to the 'set msg.payload' node, and another callout points to the 'change' node configuration panel on the right.

**3つの変数を代入**

**プルダウンメニューから「msg.」を選択**

(2-9) <http://localhost:1880/worldmap>を開くと、世界地図が表示され、宇宙ステーションの位置を確認できる



- ノードが所属するカテゴリを変更

デフォルトの「機能」カテゴリ以外にノードを所属させる場合に指定する

例) 「分析」カテゴリを指定

```
> node-red-nodegen swagger.yaml --category analysis
```

オプションの  
ハイフンは2つ

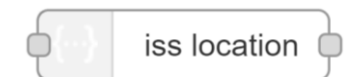


- ノードの色を変更

--color オプションに16進数で色を指定する

例) ノードを白色に設定

```
> node-red-nodegen swagger.yaml --color FFFFFFFF
```

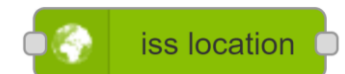


- ノードのアイコンを変更

--icon オプションにPNGファイルやNode-RED標準アイコン名を指定する

例) 地球のアイコンを設定

```
> node-red-nodegen swagger.yaml --icon white-globe
```



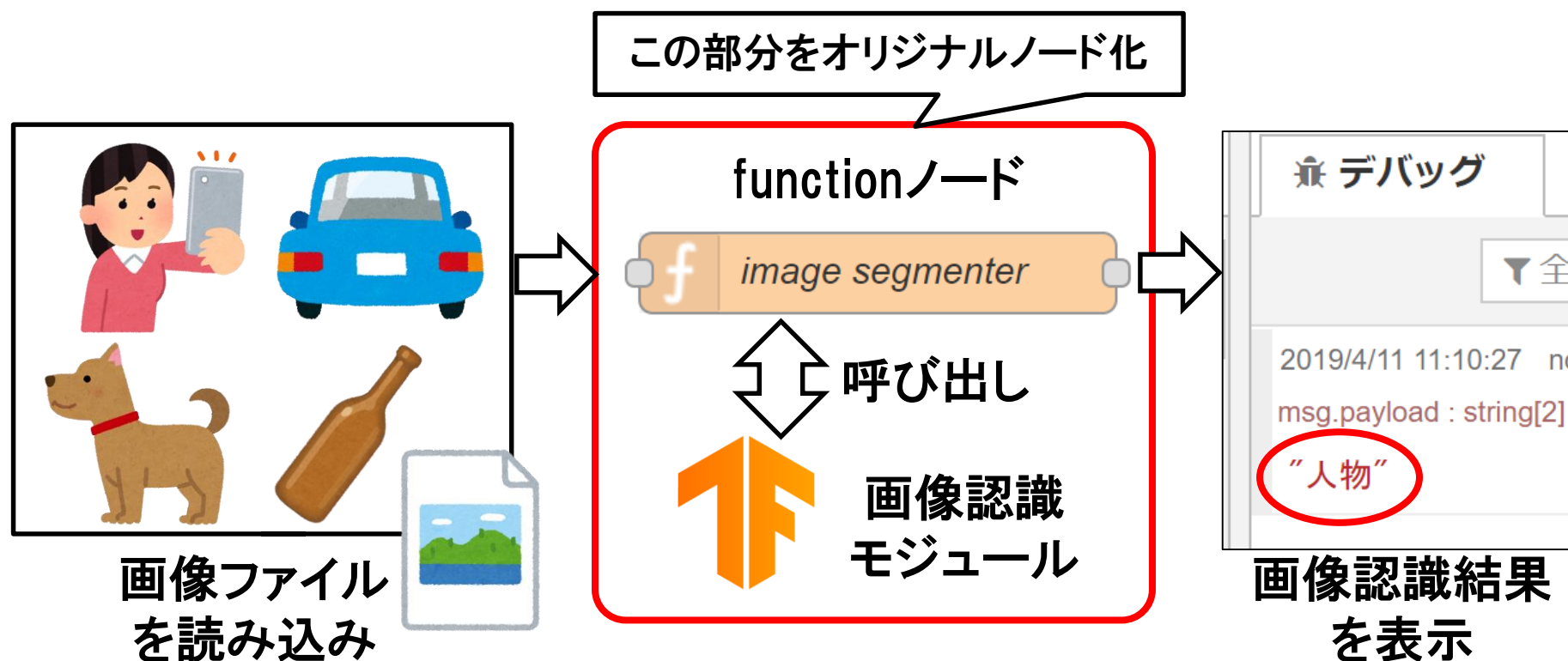




## ハンズオン2: functionノードからオリジナルノードを作成

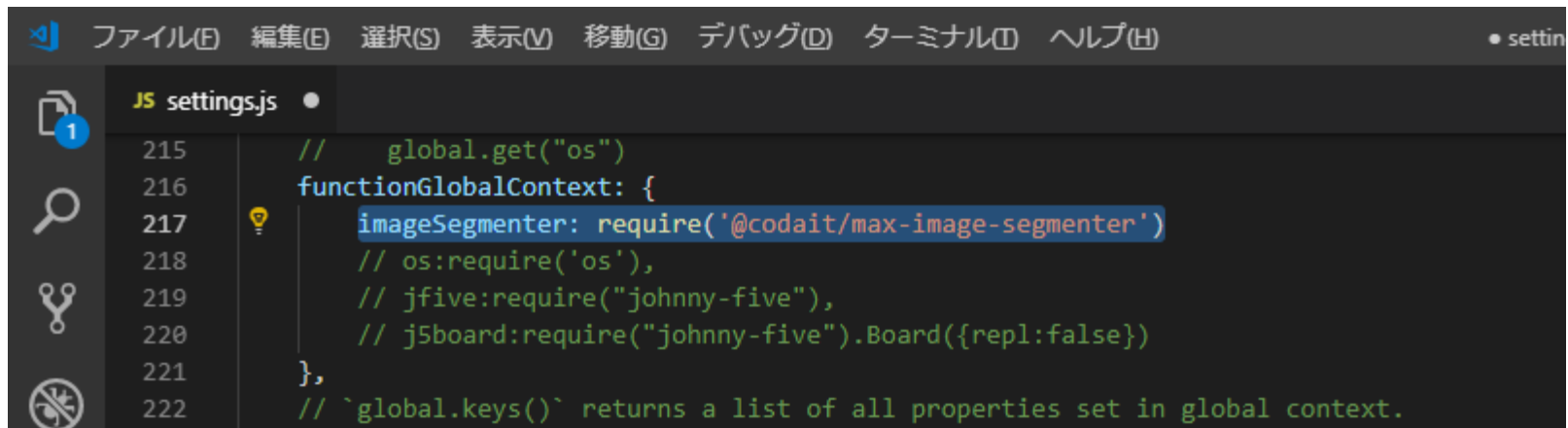
## 画像認識モジュールと連携させたfunctionノードからオリジナルノード作成

- (1) functionノードで画像認識モジュールを利用するための設定
- (2) 画像認識モジュールを利用するfunctionノード、フローを作成
- (3) functionノードからオリジナルノードを自動生成



- (1-1) テキストエディタでNode-REDの設定ファイル  
(C:¥Users¥<ユーザ名>¥.node-red¥settings.js)を開く
- (1-2) 216行目辺りのfunctionGlobalContextのブロック内に  
以下の通り画像認識モジュールのrequire文を追加し、保存

```
functionGlobalContext: {  
  imageSegmenter: require('@codait/max-image-segmenter')  
}
```



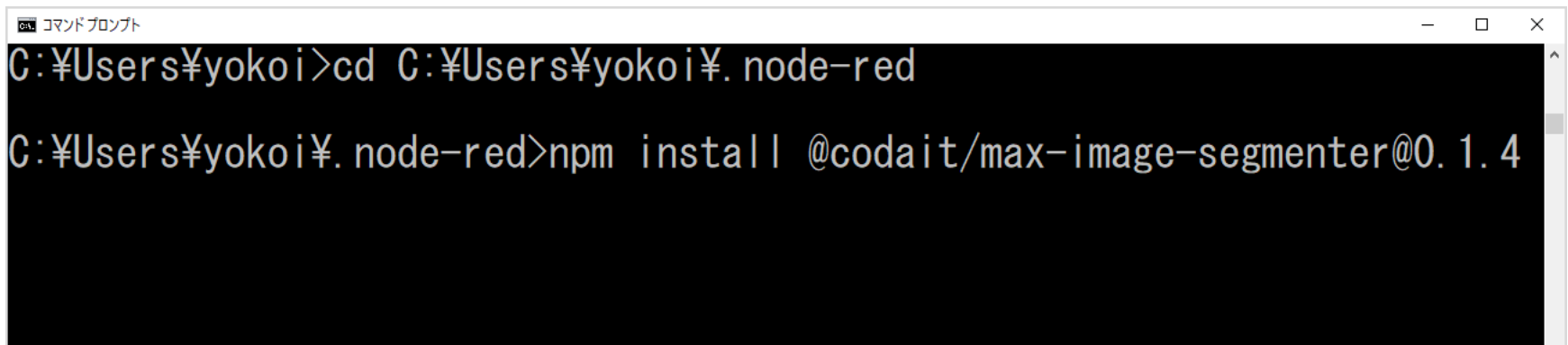
Visual Studio Codeのスクリーンショット

## (1-3) コマンドプロンプト上でNode-REDのホームディレクトリに移動

```
> cd C:¥Users¥<ユーザ名>¥.node-red
```

## (1-4) 画像認識モジュールのインストール

```
> npm install @codait/max-image-segmenter@0.1.4
```



```
コマンド プロンプト
C:¥Users¥yokoi>cd C:¥Users¥yokoi¥.node-red
C:¥Users¥yokoi¥.node-red>npm install @codait/max-image-segmenter@0.1.4
```

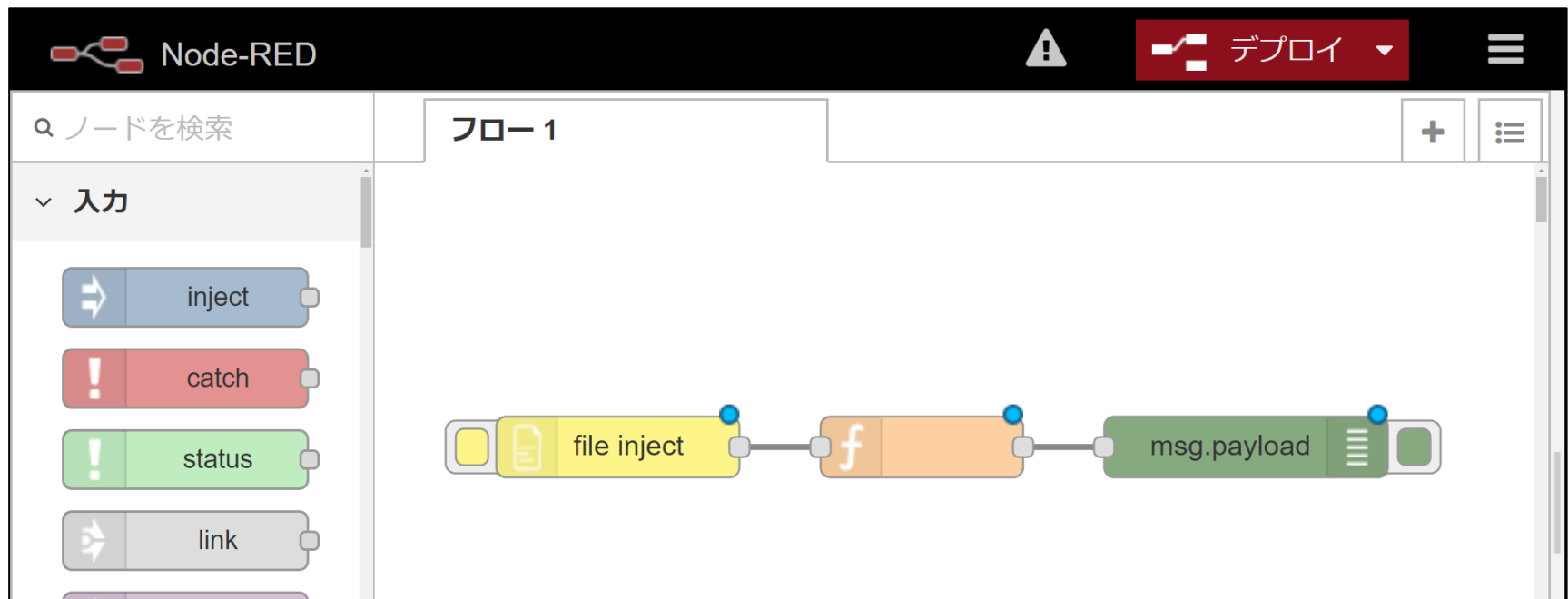
※参考: サードパーティモジュールの利用手順 <https://nodered.jp/docs/writing-functions>

(2-1) Node-REDを再起動(node-redコマンドをCtrl+cで終了後、再実行)

> node-red

(2-2) ブラウザからフローエディタ(<http://localhost:1880>)へアクセス

(2-3) file-inject、function、debugノードを順に接続し、フロー作成



## (2-4) functionノードのプロパティUIに名前、JavaScriptコードを入力

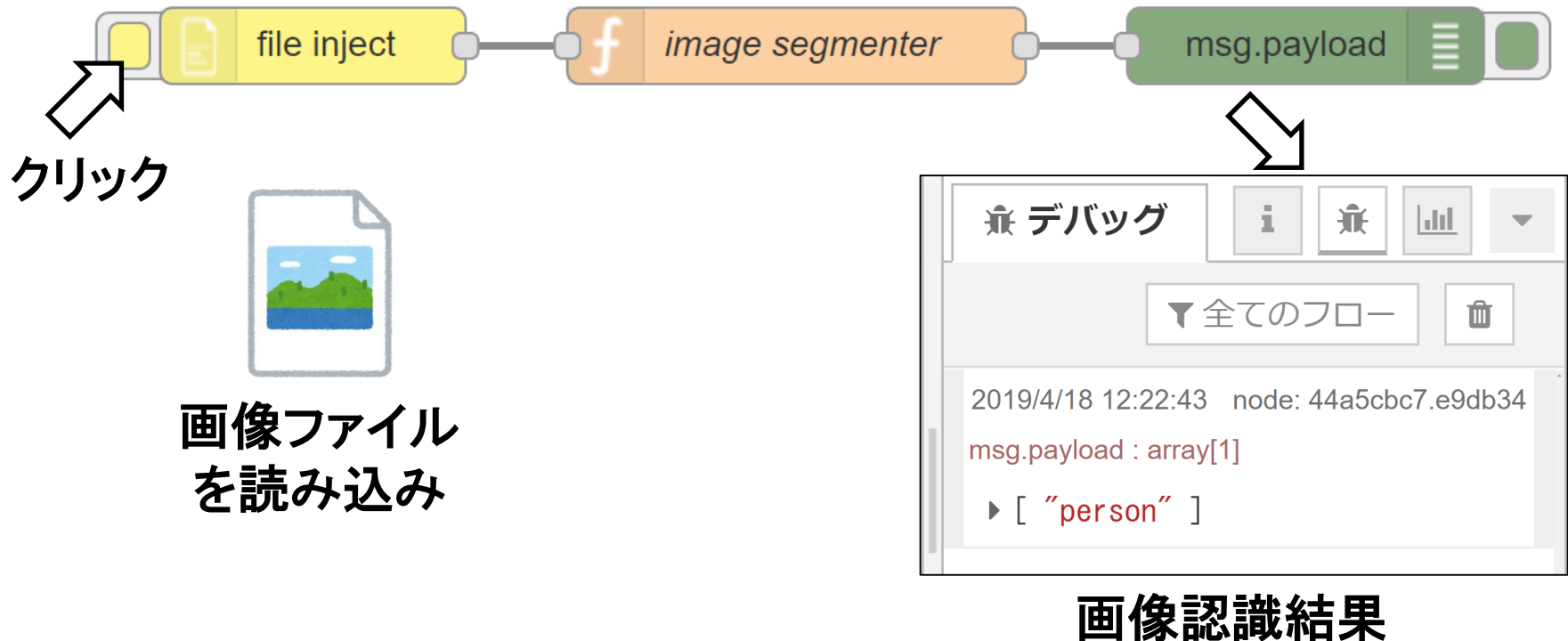
#	項目	設定値
1	名前	image segmenter
2	JavaScript コード	<pre>var imageSegmenter = global.get('imageSegmenter'); imageSegmenter.predict(msg.payload).then(function (response) {   msg.payload = response.objectsDetected;   node.send(msg); });</pre>

※ハンズオンのリポジトリにimage-segmenter.jsというファイル名でコードを置いてあります



functionノードのプロパティUIのスクリーンショット

- (2-5) Node-REDフローエディタのデプロイボタンをクリックした後、  
file-injectノードのボタンを押し、画像ファイルをアップロード  
-> デバッグタブに画像認識結果が出力される



- (3-1) functionノードのノードプロパティ上にある「本」アイコンをクリックし、  
「ライブラリへ保存」を選択  
-> JavaScriptのソースコードが  
C:\Users\<ユーザ名>\.node-red\lib\functions\へ保存される



- (3-2) コマンドプロンプト上でCtrl+cを押し、Node-REDを終了



(3-3) コマンドプロンプトを開き、ホームディレクトリに移動

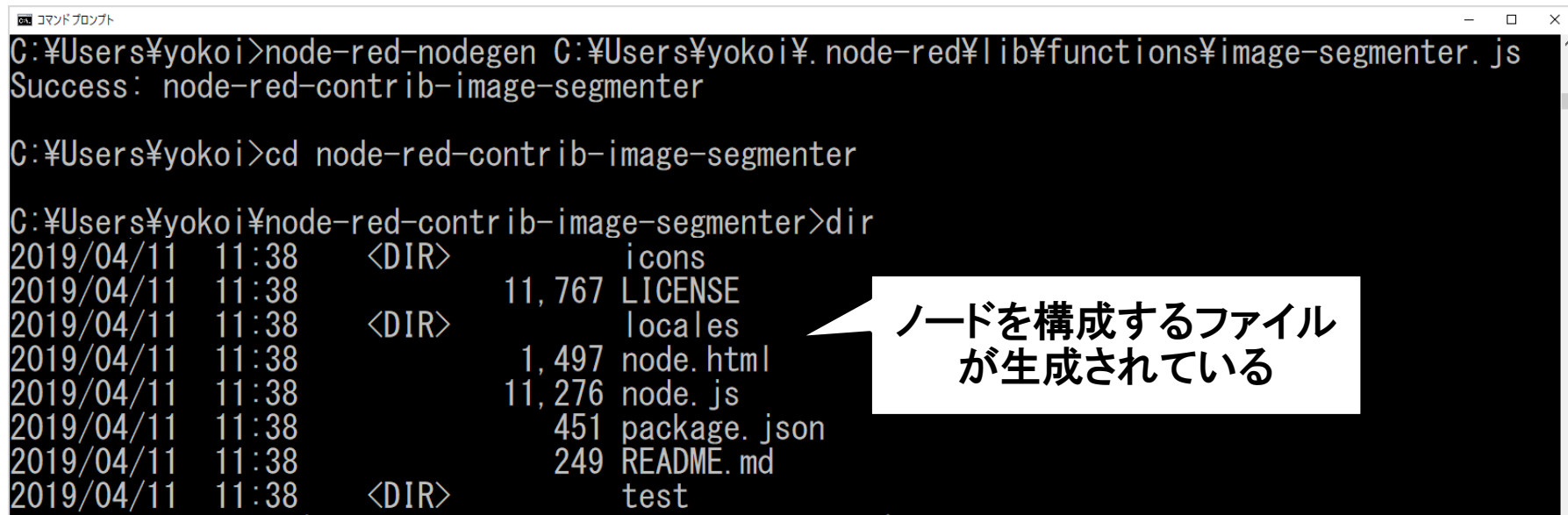
```
> cd C:\Users\<ユーザ名>
```

(3-4) Node generatorを用いてfunctionノードからノードを自動生成

```
> node-red-nodegen C:\Users\<ユーザ名>\.node-red\lib\functions\image-segmenter.js
```

(3-5) 生成したノードのディレクトリにカレントディレクトリを変更

```
> cd node-red-contrib-image-segmenter
```



```
コマンド プロンプト
C:\Users\yokoi>node-red-nodegen C:\Users\yokoi\.node-red\lib\functions\image-segmenter.js
Success: node-red-contrib-image-segmenter

C:\Users\yokoi>cd node-red-contrib-image-segmenter

C:\Users\yokoi\node-red-contrib-image-segmenter>dir
2019/04/11  11:38    <DIR>          icons
2019/04/11  11:38             11,767 LICENSE
2019/04/11  11:38    <DIR>          locales
2019/04/11  11:38             1,497 node.html
2019/04/11  11:38             11,276 node.js
2019/04/11  11:38              451 package.json
2019/04/11  11:38              249 README.md
2019/04/11  11:38    <DIR>          test
```

ノードを構成するファイル  
が生成されている

## (3-6) シンボリックリンク作成の準備

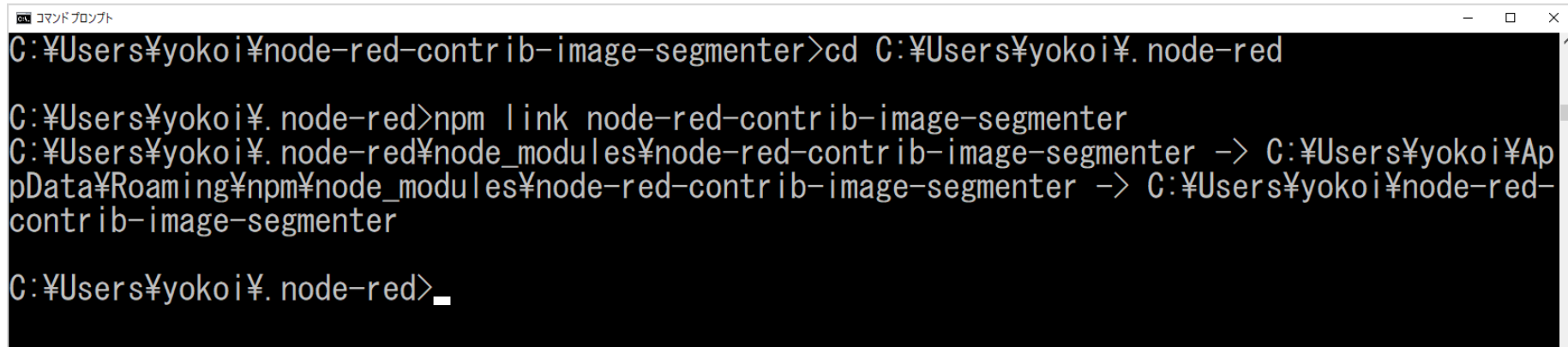
```
> npm link
```

## (3-7) Node-REDのホームディレクトリに移動

```
> cd C:¥Users¥<ユーザ名>¥.node-red
```

## (3-8) シンボリックリンクを作成

```
> npm link node-red-contrib-image-segmenter
```



```
コマンド プロンプト
C:¥Users¥yokoi¥node-red-contrib-image-segmenter>cd C:¥Users¥yokoi¥.node-red

C:¥Users¥yokoi¥.node-red>npm link node-red-contrib-image-segmenter
C:¥Users¥yokoi¥.node-red¥node_modules¥node-red-contrib-image-segmenter -> C:¥Users¥yokoi¥AppData¥Roaming¥npm¥node_modules¥node-red-contrib-image-segmenter -> C:¥Users¥yokoi¥node-red-contrib-image-segmenter

C:¥Users¥yokoi¥.node-red>_
```

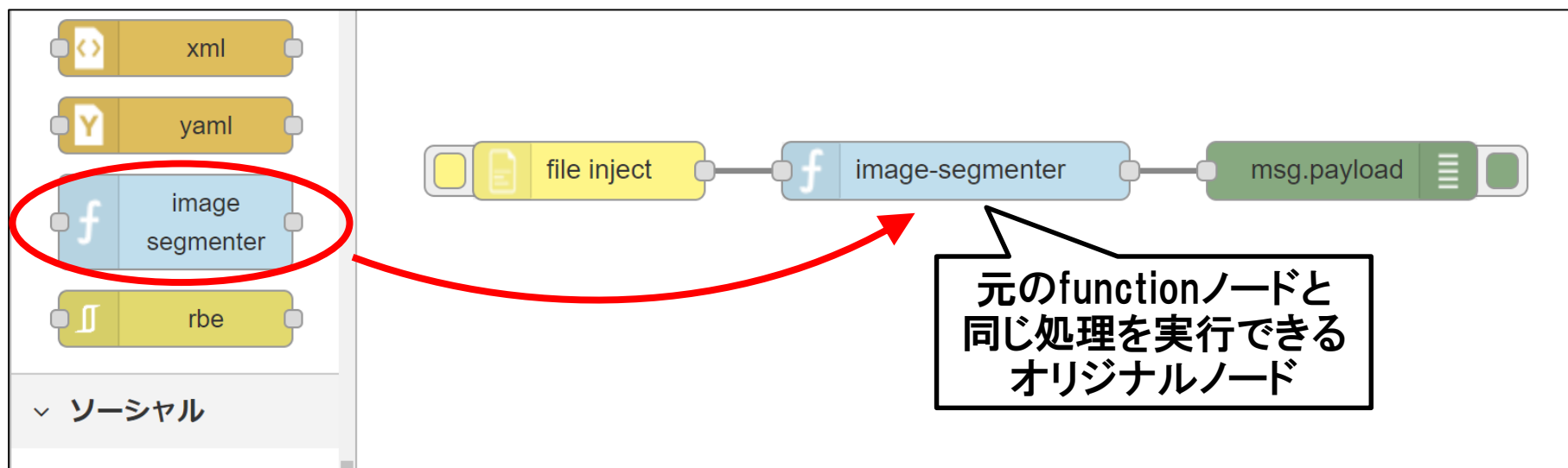
## (3-9) Node-REDを起動

**> node-red**

## (3-10) フローエディタ(<http://localhost:1880>)にアクセス(またはリロード)

-> パレットの「機能」カテゴリの中にimage segmentatorノードが登場

## (3-11) 元のfunctionノードと置きかえ、同じ処理を実行できることを確認



オリジナルノードをフローライブラリに公開する場合、  
settings.jsの修正なくノードを使える様にするコードの修正が必要

- ノードのJavaScriptファイル(node-red-contrib-image-segmenter¥node.js)に  
画像認識モジュールを読み込むコードを追加

```
sandbox.global.set('imageSegmenter',  
                    require('@codait/max-image-segmenter'));  
var context = vm.createContext(sandbox);  
try {  
  this.script = vm.createScript(functionText, {
```

vm.createContext()  
の前にコードを追加

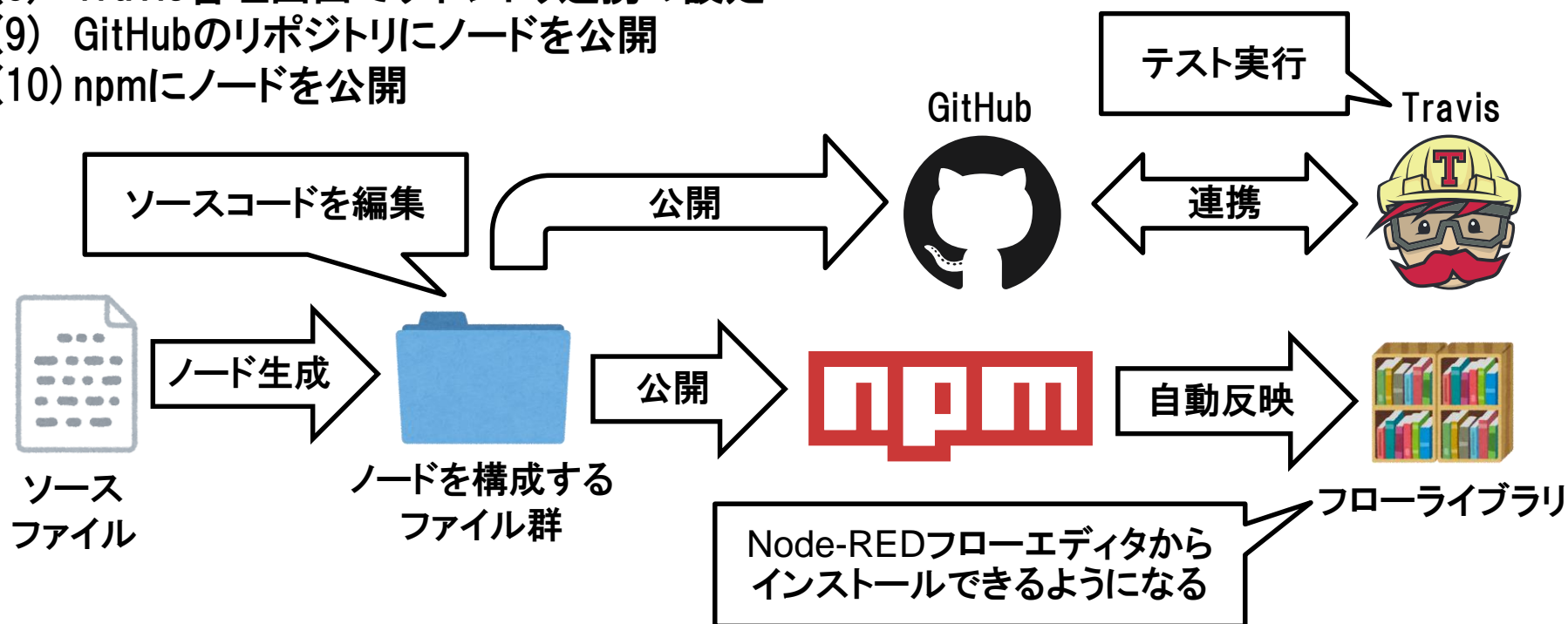
- パッケージ情報ファイル(node-red-contrib-image-segmenter¥package.json)  
の依存関係ブロックに画像認識モジュールを記載

```
"keywords": [ ... ],  
"dependencies": {  
  "@codait/max-image-segmenter": "0.1.4"  
},  
"devDependencies": {
```

依存関係を追加

# 補足: フローライブラリにノードを登録する手順

- (1) Node generatorで「--keywords node-red」オプションでキーワードを指定してノード生成
- (2) 情報タブに表示するノードの使い方をHTMLファイル(node.html)に記載
- (3) ノードの概要やインストール方法のドキュメント(README.md)を記載
- (4) ノードプロパティUIの日英メッセージのファイル(locales/<言語>/node.json)を編集
- (5) ノードのテストケース(test/node\_spec.js)を追加
- (6) Travis設定ファイル(.travis.yml)を追加
- (7) packages.jsonにGitHubリポジトリのURLを追加
- (8) Travis管理画面でリポジトリ連携の設定
- (9) GitHubのリポジトリにノードを公開
- (10) npmにノードを公開





**最後に**

- オリジナルノードを公開し、Node-REDコミュニティを盛り上げましょう
- ノード開発のサポートが必要な場合は、Node-RED公式Slackの#nodegenチャンネルにご質問をお願いします(日本語可)

Node-RED Library

Find new nodes, share your flows and see what's popular

Search library

☐ flows ☒ nodes

Sort by:  
☒ recent  
☐ downloads  
☐ rating

1948 of 3,833 things

ライブラリに登録されているノードはもうすぐ2000個！

<b>node-red-contrib-ssh-client</b> A Node-RED node for running remote ssh command. v0.0.1 node	<b>node-red-contrib-lets-encrypt</b> A Node-RED node for exposing an https server with let's encrypt. v0.0.3 node	<b>node-red-contrib-elasticsearch-jupalc</b> elasticsearch Wrapper for NodeRed v1.1.1 node
--	---	--

フローライブラリ (<https://flows.nodered.org>)

**END**



**Node-REDのノード開発ハンズオン**

2019/04/22

日立製作所 中央研究所

**横井 一仁**



**HITACHI**  
Inspire the Next 

- IBMは、International Business Machines Corporationの登録商標です。
- Googleは、Google LLCの登録商標です。
- Xcodeは、Apple Inc.の登録商標です。
- GitHubは、GitHub Incorporatedの登録商標です。
- npmは、npm Incorporatedの登録商標です。
- Slackは、Slack Technologies Inc.の登録商標です。