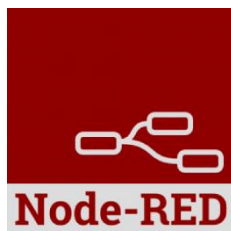
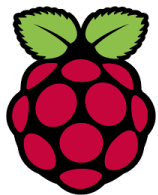


# WLOラズパイ倶楽部

## Node-REDでIoTにチャレンジ



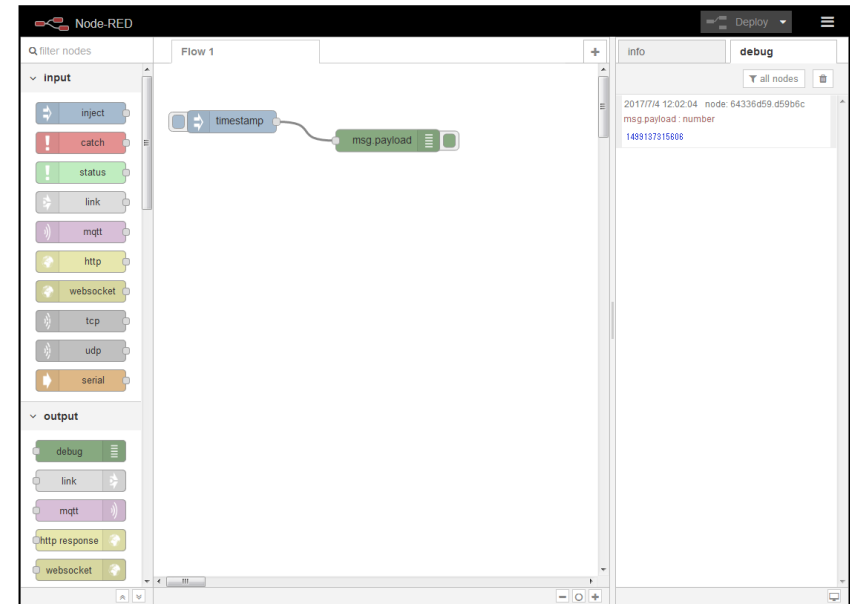
2017年7月27日

- ・IBMが開発し、JS Foundationに寄贈した「IoT向けプラットフォーム」

ライセンスは「Apache 2.0 License」

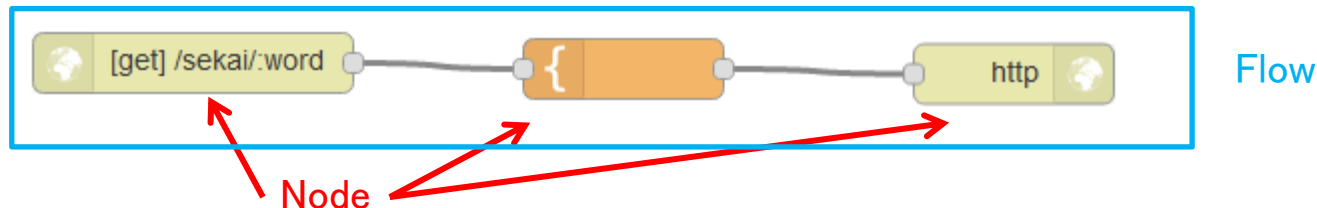
- ・ブラウザ上で開発/デプロイ

WebブラウザでFlowEditorにアクセスすることで開発を開始。



- ・ビジュアルプログラミング環境

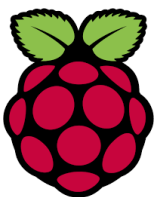
機能モジュールである「Node」を並べ、接続することで処理の「Flow」を定義。  
ブラウザ上の「Deploy」でそのままサーバにデプロイ。  
そのまま実行できる。



## ・移植性

Node.jsベースで開発されており、さまざまなOS環境で動作する。

## 組み込み環境



## PC/サーバ

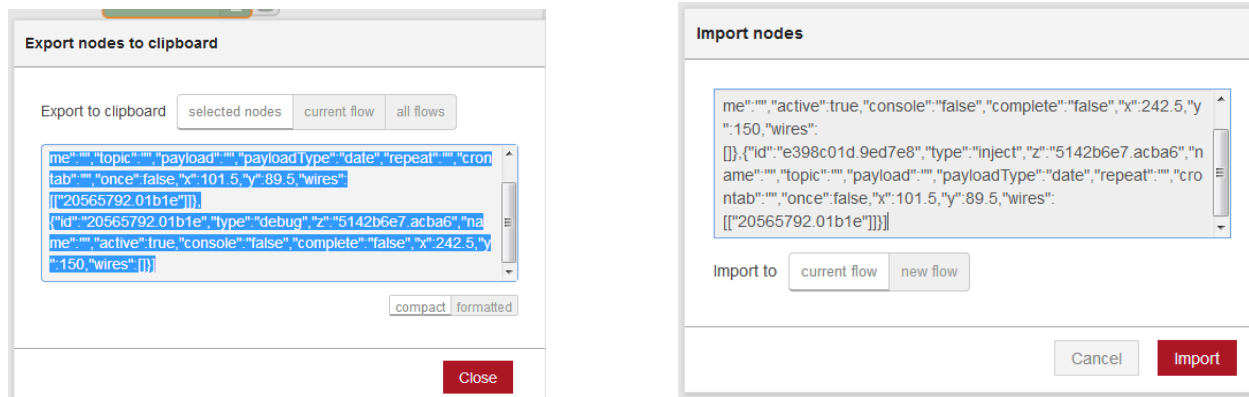


## パブリッククラウド

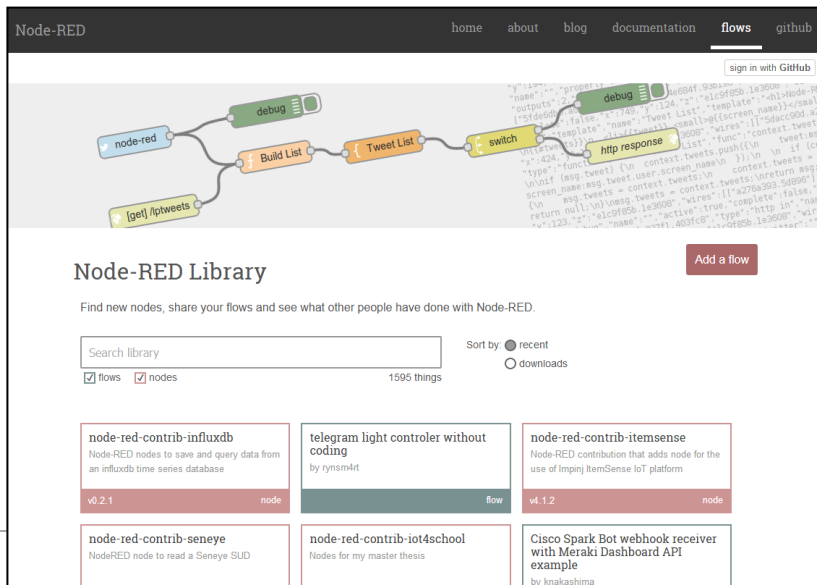


## ・再利用性

つくったFlowはJSONフォーマットでFlowEditorへExport/Importできる。  
⇒他の人がつくったFlowを再利用できる



## ・拡張性



Node-RED Libraryで  
公開されている  
CustomNodeを  
取り込むことで、  
各種Webサービスや  
各社サービスの機能を  
FlowEditorに組み込むことができる。

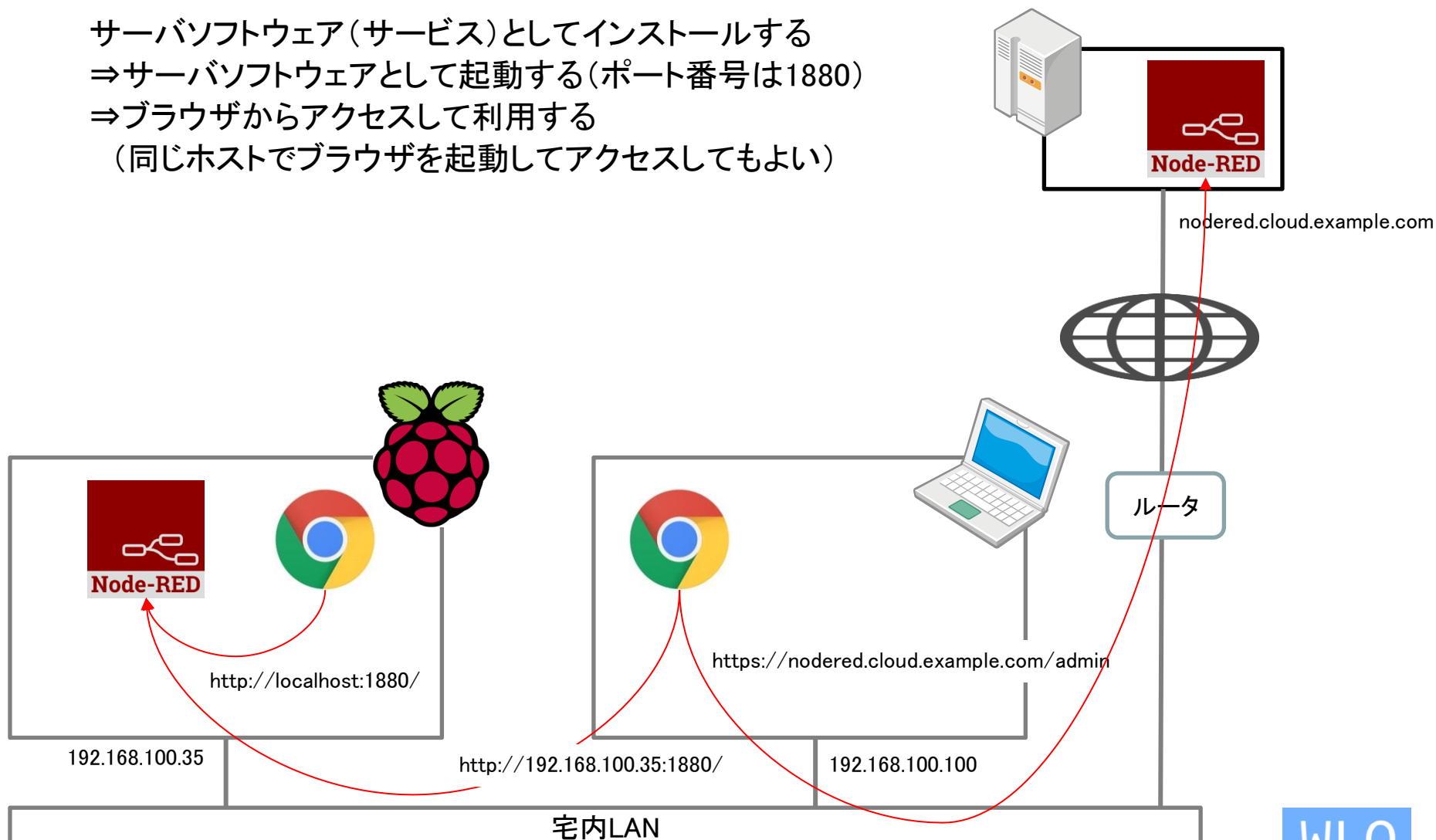
## ・Node-REDはサーバソフトウェア

サーバソフトウェア(サービス)としてインストールする

⇒サーバソフトウェアとして起動する(ポート番号は1880)

⇒ブラウザからアクセスして利用する

(同じホストでブラウザを起動してアクセスしてもよい)

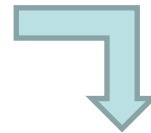
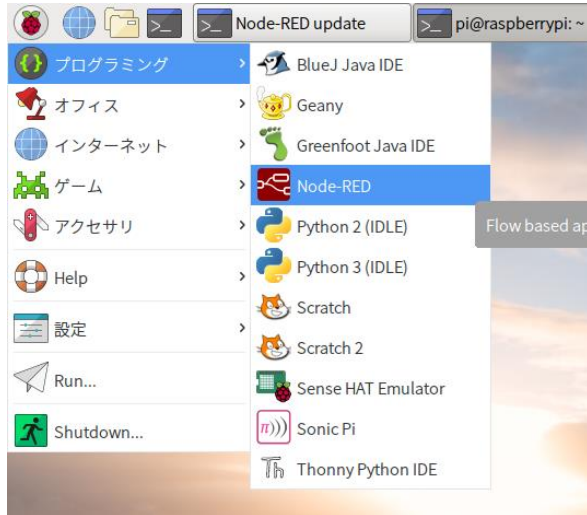


- ・体験用環境(クラウド環境に導入済み)

お渡ししたURLにアクセスして、ログインしてお使いください

## ・Raspberry Piではじめる

Raspbian 2015-11-23版より、Node-REDが標準でインストールされている。



コンソールが開いて  
サービスが起動する

```
Node-RED console
ファイル(F) 編集(E) タブ(T) ヘルプ(H)

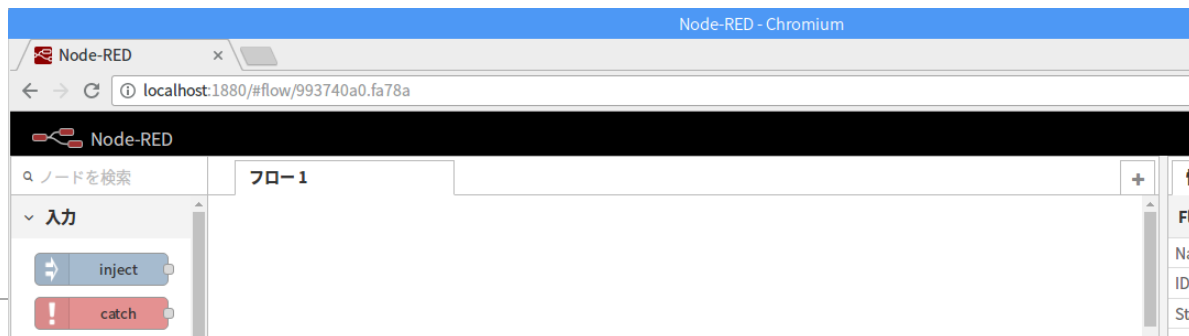
Start Node-RED

Once Node-RED has started, point a browser at http://192.168.0.20:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use node-red-stop           to stop Node-RED
Use node-red-start          to start Node-RED again
Use node-red-log             to view the recent log output
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

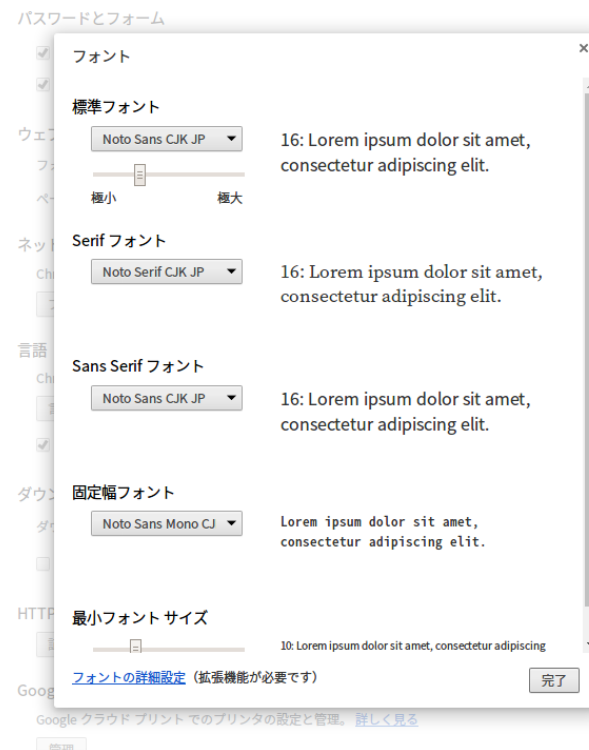
Starting as a systemd service.
Started Node-RED graphical event wiring tool..
27 Jul 08:48:04 - [info]
Welcome to Node-RED
=====
27 Jul 08:48:04 - [info] Node-RED version: v0.17.5
27 Jul 08:48:04 - [info] Node.js version: v6.11.1
27 Jul 08:48:04 - [info] Linux 4.9.35-v7+ arm LE
27 Jul 08:48:06 - [info] Loading palette nodes
27 Jul 08:48:10 - [info] Settings file : /home/pi/.node-red/settings.js
```



ブラウザで  
http://localhost:1880/  
にアクセス

## ・標準ブラウザ「Chromium」(Chrome)でアクセス

標準ブラウザ「Chromium」だと  
フォントの設定によっては、  
編集時に文字ずれが発生する  
⇒Notoフォントがおすすめ



## ・(参考) Notoフォントの導入スクリプト

```
$ wget https://git.io/wlonoto
$ bash wlonoto
```

<https://github.com/WLO-RaspiClub/Node-RED/blob/master/noto.sh>



## ・Raspberry PiにおけるNode-REDの更新

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
pi@raspberrypi:~$ update-nodejs-and-nodered  
  
This script will remove any pre-installed versions of node.js and Node-RED  
and replace them with node.js 6.x LTS (boron) and the latest Node-RED from Npm.  
To do this it runs commands as root - please satisfy yourself that this will  
not damage your Pi, or otherwise compromise your configuration.  
  
Doing this may also be 'a bad thing' if you have installed lots of extra nodes.  
Especially if they have any native binary component. Some nodes in your  
~/node-red directory will probably need to be re-installed afterwards, some  
may need you to run npm update, and some may require you to run npm rebuild.  
  
There may be a period of frustration ahead to get back to where you were...  
  
Are you really, really sure you want to do this ? y  
rm: '/var/log/nodered-install.log' を削除できません: そのようなファイルやディレ  
クトリはありません  
  
This can take 20-30 minutes on a Pi 1 - please wait.  
  
Stop Node-RED ✓  
Remove old version of node.js ✓  
Install node.js for Armv7 ✓ Node v6.11.1 Npm 3.10.10  
Clean npm cache ✓  
Install Node-RED core ✓  
Install extra nodes ✓  
Install serialport node ✓  
Npm rebuild existing nodes ✓  
Add menu shortcut ✓  
Update systemd script ✓  
Update update script ✓  
  
Any errors will be logged to /var/log/nodered-install.log  
  
All done.  
You can now start Node-RED with the command node-red-start  
or using the icon under Menu / Programming / Node-RED  
Then point your browser to localhost:1880 or http://{your_pi_ip-address}:1880  
  
Started 2017年 7月 27日 木曜日 08:34:55 JST - Finished 2017年 7月 27日 木  
曜日 08:45:12 JST  
pi@raspberrypi:~$
```

Raspbianで標準サポートされている  
ミドルウェアが古い場合があり、  
そのままでは安定して動作しない。

⇒更新用スクリプト

「update-nodejs-and-nodered」を用いて  
Node.jsとNode-REDを最新版に更新する  
(所要時間約12分)

## ・Raspberry PiにおけるNode-REDの自動起動設定・解除

自動起動設定

「`sudo systemctl enable nodered.service`」

自動起動解除

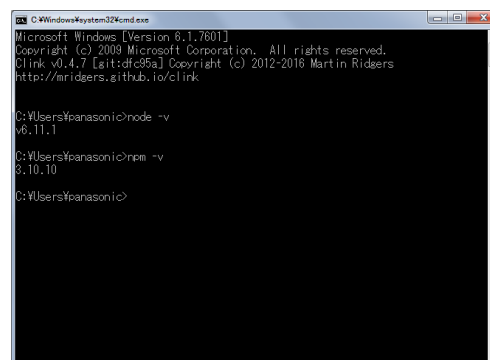
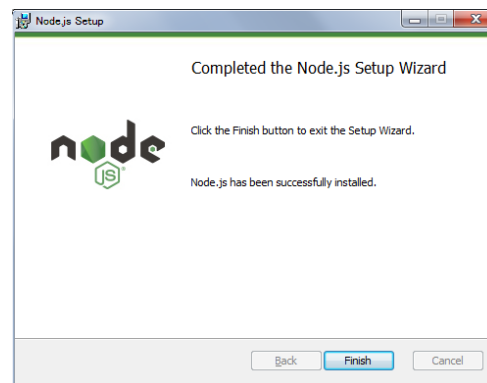
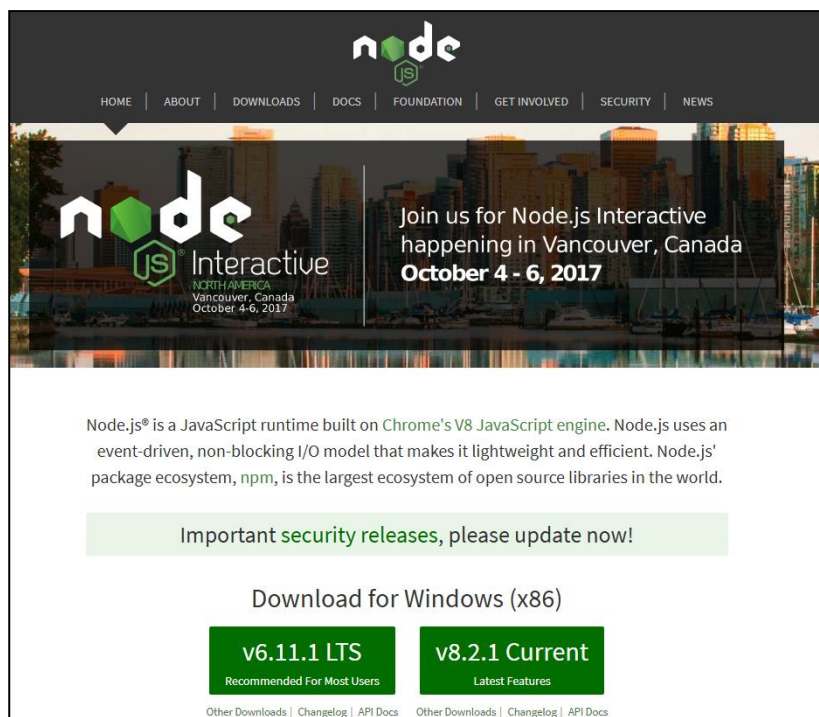
「`sudo systemctl disable nodered.service`」

Node-REDサービスの再起動

「`sudo systemctl restart nodered.service`」

## ・必要なミドルウェアの導入

Node.jsのLTS(Long Term Support)版をインストーラでインストールする  
(2017/7/27現在 v6.11.1)  
標準でnpm 3.xが導入される



## ・npmでNode-REDの導入

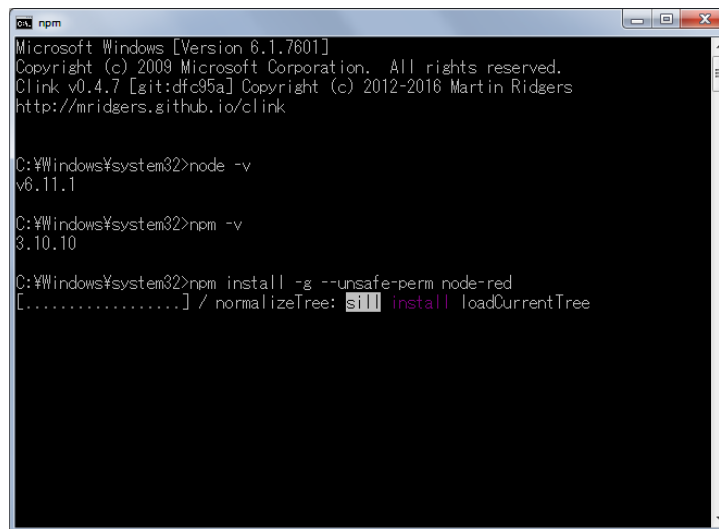
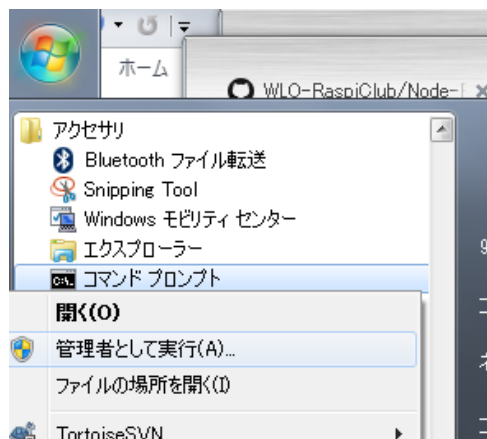
管理者権限が必要

・Linux/macOSの場合はsudoを使って管理者で

「sudo npm install -g --unsafe-perm node-red」

・Windowsの場合はコマンドプロンプトを「管理者として実行」で

「npm install -g --unsafe-perm node-red」



・起動は「node-red」

・ブラウザで「http://localhost:1880」にアクセスする

## ・dockerでの導入

<https://hub.docker.com/r/nodered/node-red-docker/>

The screenshot shows the Docker Hub repository page for `nodered/node-red-docker`. The page header includes a search bar, "Explore", "Help", "Sign up", and "Sign in" links. Below the header, it indicates the repository is "PUBLIC | AUTOMATED BUILD" and shows the repository name with a star icon. The "Last pushed" time is "2 days ago". The main content area is divided into two columns. The left column contains a "Short Description" (Node-RED Docker images), a "Full Description" (Node-RED-Docker) with a detailed explanation of the project and a code block for the `docker run` command, and a "Source Repository" link. The right column contains a "Docker Pull Command" (docker pull nodered/node-red-docker), an "Owner" section (nodered), and a "Source Repository" link.

PUBLIC | AUTOMATED BUILD

[nodered/node-red-docker](#) ☆

Last pushed: 2 days ago

Repo Info Tags Dockerfile Build Details

Short Description

Node-RED Docker images.

Full Description

**Node-RED-Docker**

This project describes some of the many ways Node-RED can be run under Docker. Some basic familiarity with Docker and the [Docker Command Line](#) is assumed.

This project also provides the build for the `nodered/node-red-docker` container on [DockerHub](#).

To run this directly in docker at it's simplest just run

```
docker run -it -p 1880:1880 --name mynodered nodered/node-red-docker
```

Let's dissect that command...

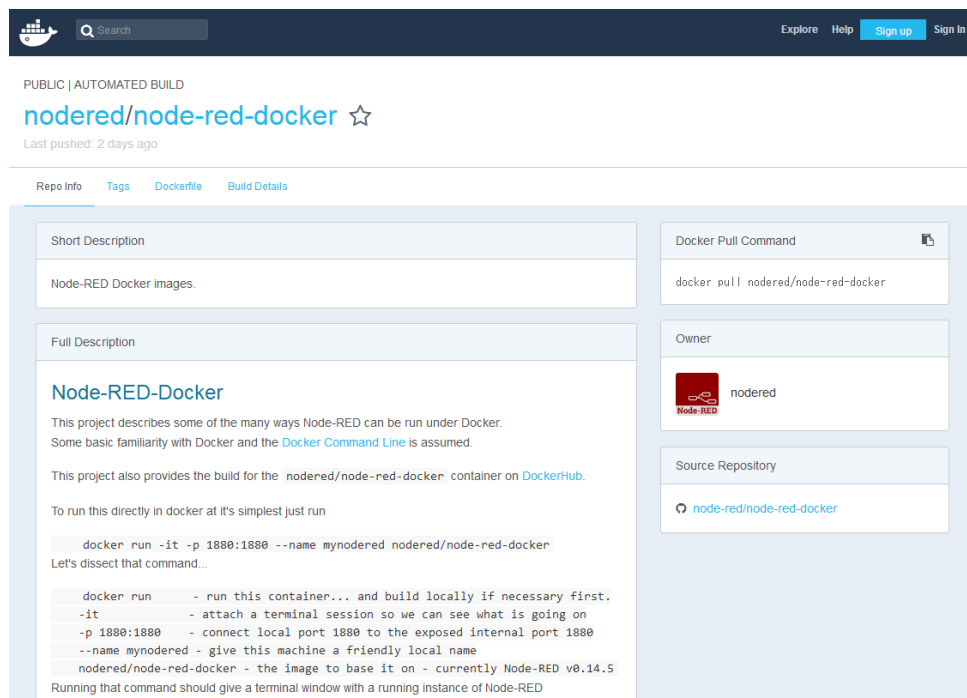
```
docker run - run this container... and build locally if necessary first.
-it - attach a terminal session so we can see what is going on
-p 1880:1880 - connect local port 1880 to the exposed internal port 1880
--name mynodered - give this machine a friendly local name
nodered/node-red-docker - the image to base it on - currently Node-RED v0.14.5
```

Running that command should give a terminal window with a running instance of Node-RED

Docker Pull Command

```
docker pull nodered/node-red-docker
```

Owner

 nodered

Source Repository

[node-red/node-red-docker](#)

## ・Node-REDサイトのインストール手順を参照

<https://nodered.org/docs/getting-started/>

Node-RED

homeaboutblogdocumentationflowsgithub

← back

Getting Started

Installation

Running

Adding Nodes

Upgrading

Creating your first flow

Creating your second flow

Docker

On a device

Raspberry Pi

BeagleBone Black

Android

Arduino

In the cloud


IBM Bluemix

SenseTecnica FRED

Amazon Web Services

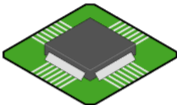
## Getting Started

This guide will help you get Node-RED installed and running in just a few minutes.




### Run locally

- Installation
- Upgrading
- Running
- Creating your first flow
- Creating your second flow
- Docker



### On a device

- Raspberry Pi
- BeagleBone Black
- Interacting with Arduino
- Android



### In the cloud

- IBM Bluemix
- SenseTecnica FRED
- Amazon Web Services
- Microsoft Azure

## ・FlowEditorの日本語化

有志によりFlowEditorの日本語化が進められているが、  
わかりにくい箇所も多い

デフォルトで導入されているので、  
ブラウザの言語設定で自動的に  
日本語リソースが使われてしまう。  
日本語リソースを削除することによって英語のFlowEditorとなる

### Linux/Raspbianの場合

/usr/lib/node\_modules/node-red

配下の

red/api/locales/ja

nodes/core/locales/ja

を削除する

### Windowsの場合

C:\Users\ユーザー\AppData\Roaming\npm\node\_modules\node-red

配下の

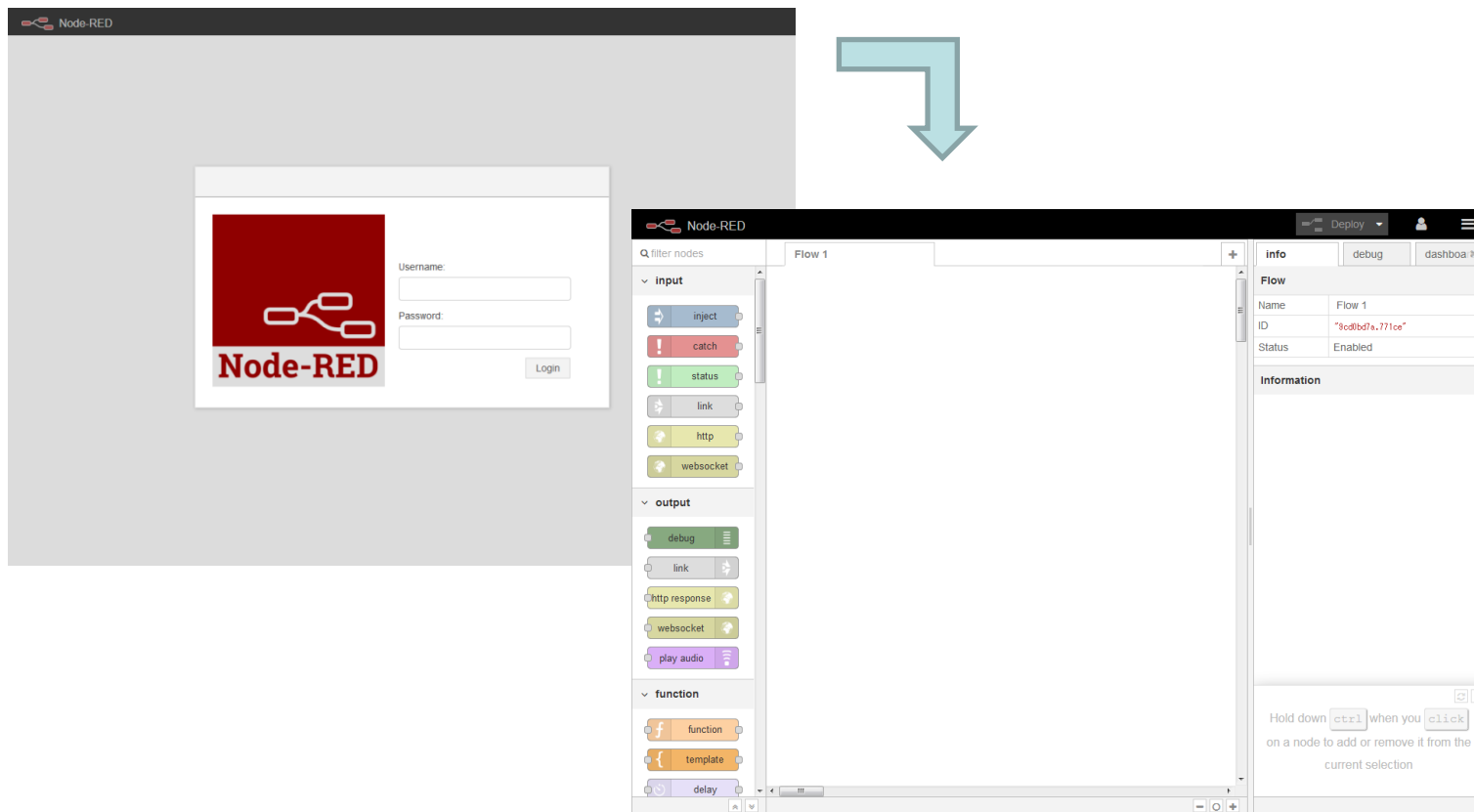
red\api\locales\ja

nodes\core\locales\ja

を削除する



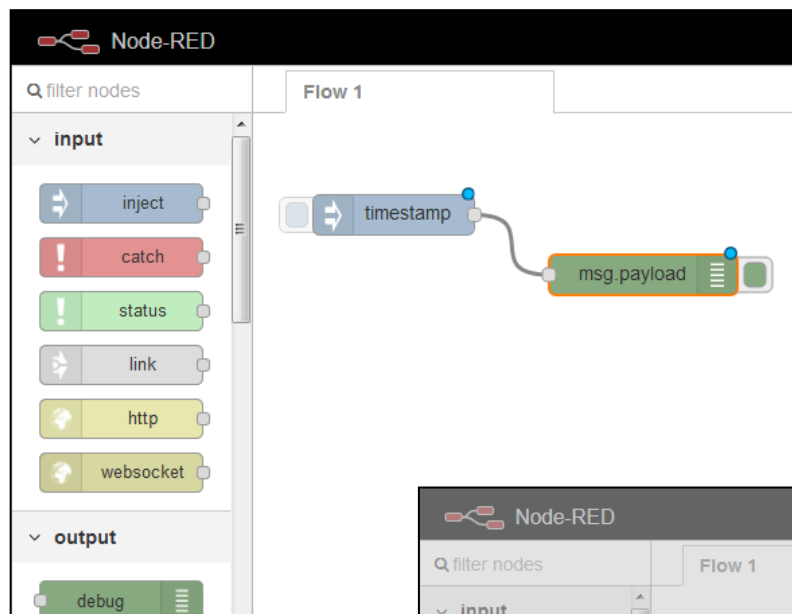
- ・クラウド環境の場合はログイン画面が表示される(ことが多い)



※自分でインストールした場合でも  
<https://nodered.org/docs/security>  
の要領でsettings.jsに設定することで、ログイン画面を出すことができる

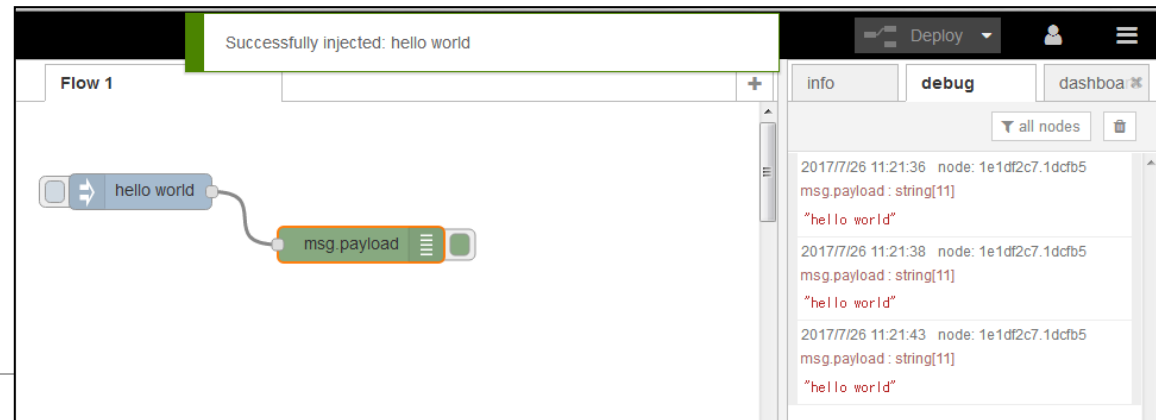
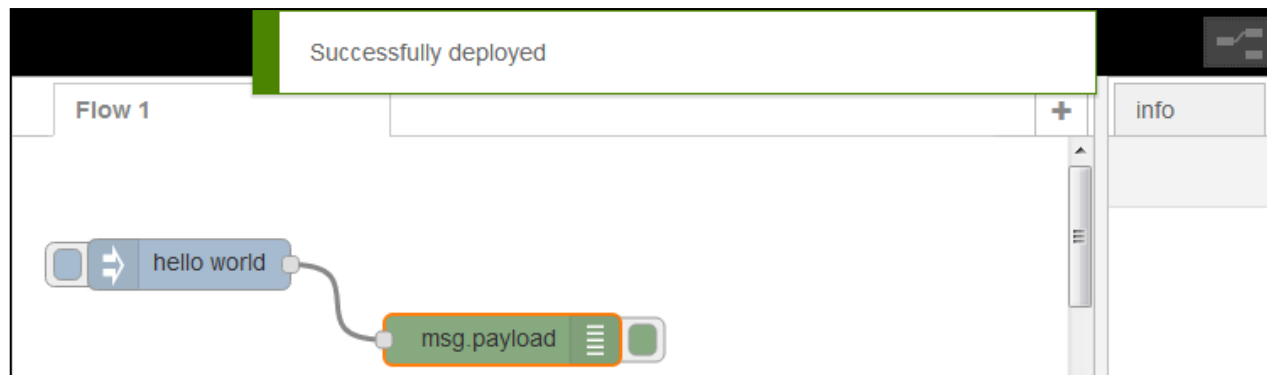


## ■最初のFlow: Hello World



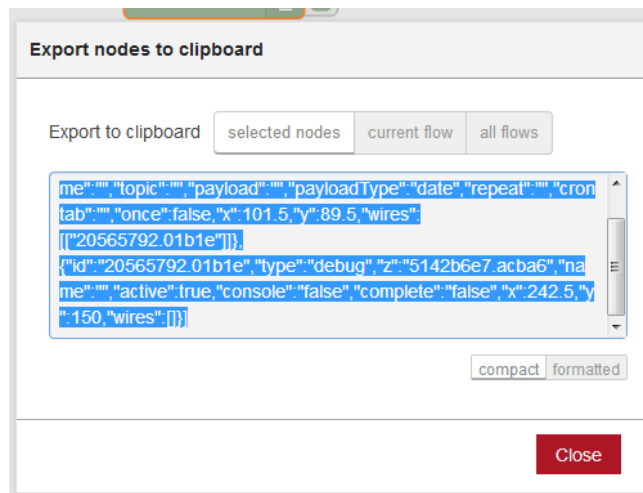
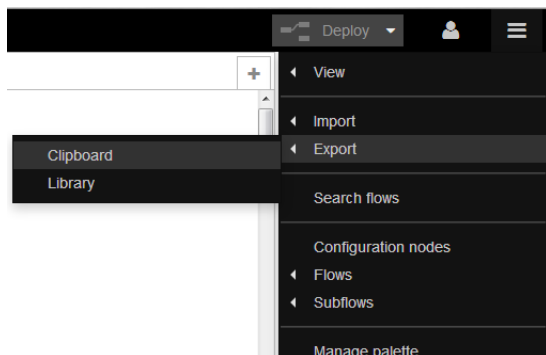
The diagram shows the Node-RED web interface with the 'Edit inject node' dialog open. The dialog has a 'Delete' button, 'Cancel' and 'Done' buttons, and a 'node properties' section. The 'Payload' field is set to 'hello world'. The 'Topic' field is empty. The 'Repeat' dropdown is set to 'none'. The 'Inject once at start?' checkbox is unchecked. The 'Name' field is set to 'Name'. A note at the bottom states: "Note: 'interval between times' and 'at a specific time' will use cron. See info box for details." The sidebar on the left shows the 'input' category with nodes like 'inject', 'catch', 'status', 'link', 'http', and 'websocket'. The 'output' category shows a 'debug' node and a 'link' node. The 'Flow 1' tab is active, and the 'msg.payload' output node is highlighted.

## ■デプロイ⇒実行

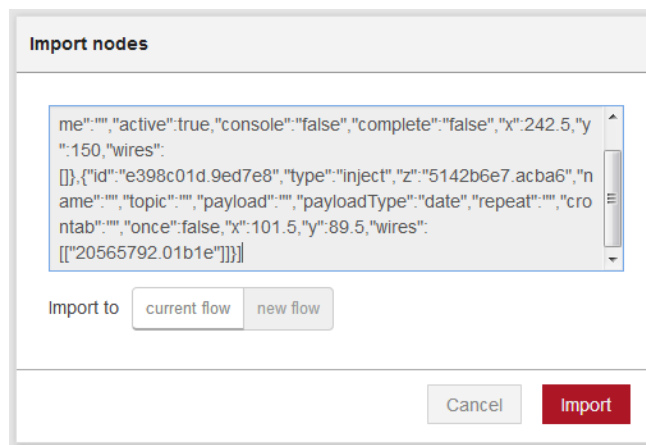
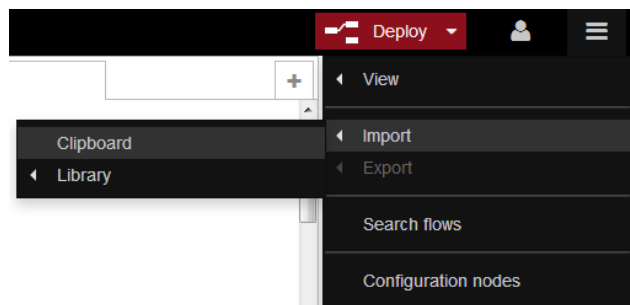


## ■ FlowのExport/Import

### ・Export



### ・Import



## ■ 静的Webページを作成

Edit http in node

Cancel Done

Method GET

URL /sekai

Name Name

Edit template node

Cancel Done

Name Name

Set property msg.payload

Template Syntax Highlight mustache

```
1 こんにちは世界！
```



## ■動的Webページを作成

Edit http in node

Cancel Done

Method GET

URL /sekai/:word

Name Name

Edit template node

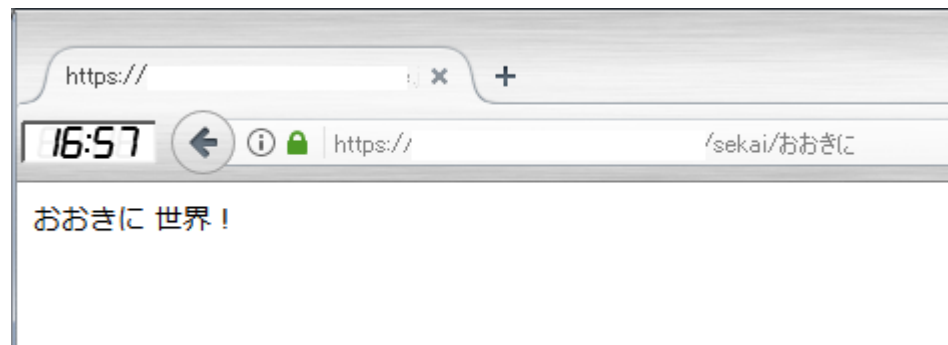
Cancel Done

Name Name

Set property msg.payload

Template Syntax Highlight mustache

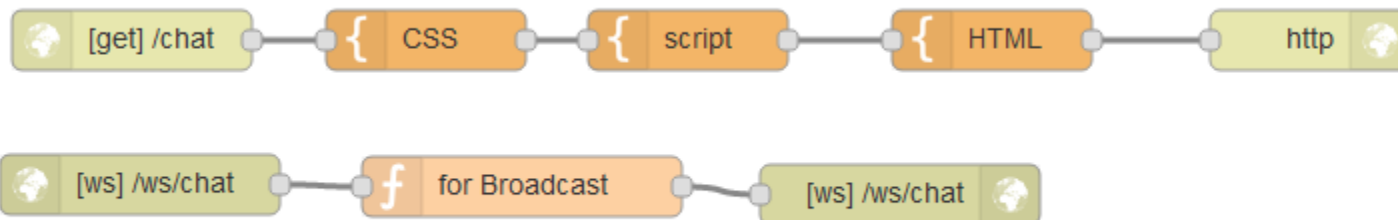
```
1 {{req.params.word}} 世界！
```



## ■WebSocketによるWebチャット

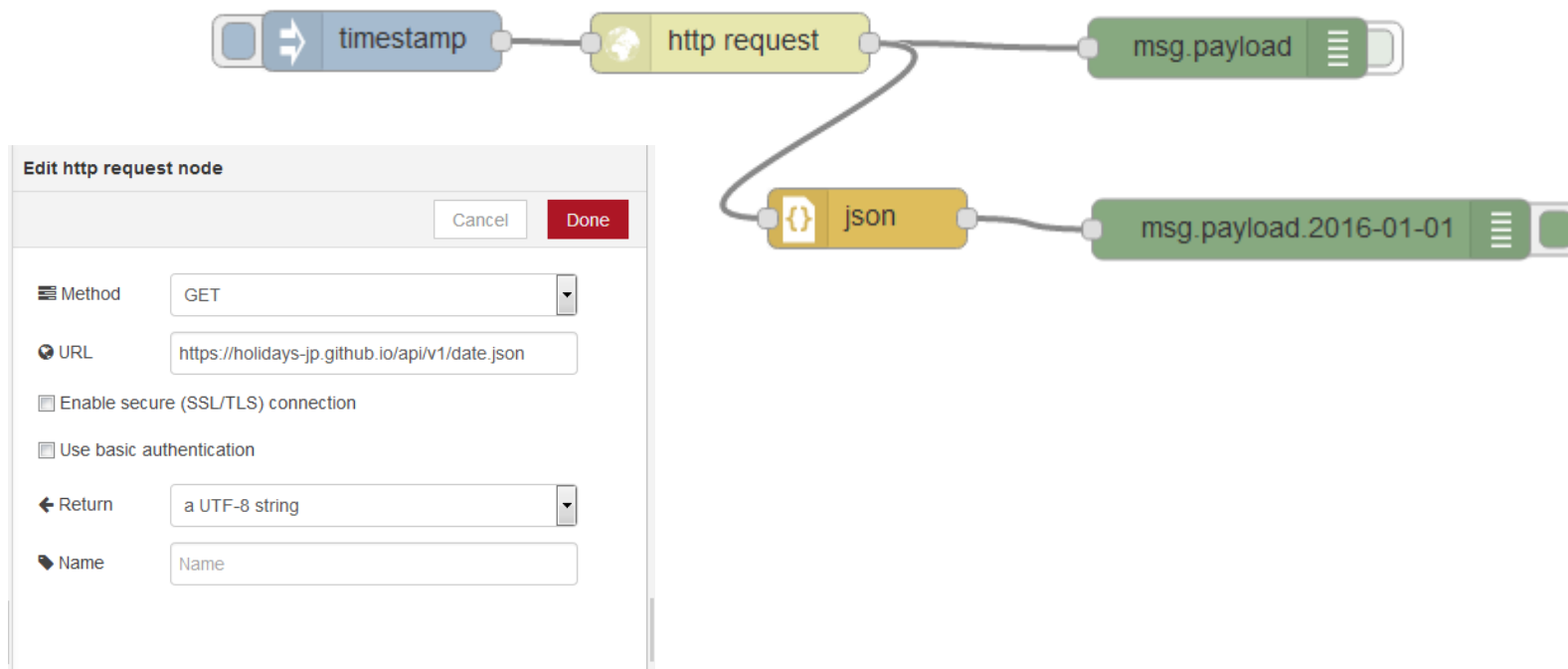
<https://github.com/WLO-RaspiClub/Node-RED/blob/master/wc.json>

<https://git.io/wlocwc>



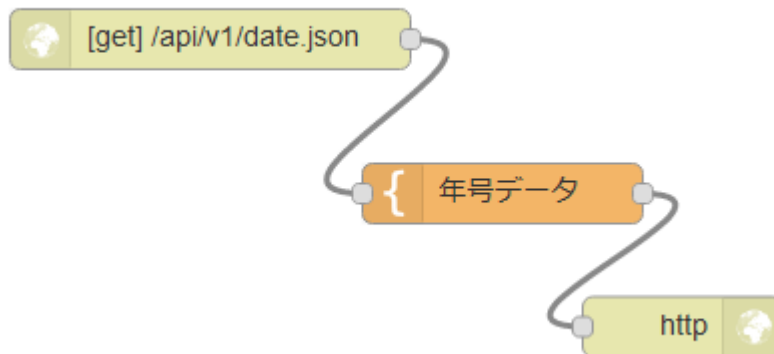
## ■外部Webサービスへのアクセス

日本の祝日を JSON / CSV 形式で返す API <https://holidays-jp.github.io/>  
(解説) <http://qiita.com/matsuoshi/items/7c19e7dcf404b7d921d6>



<https://github.com/WLO-RaspiClub/Node-RED/blob/master/nengo.json>  
<https://git.io/wlonengo>

## ■課題: 日本の祝日APIをNode-REDでつくってみる

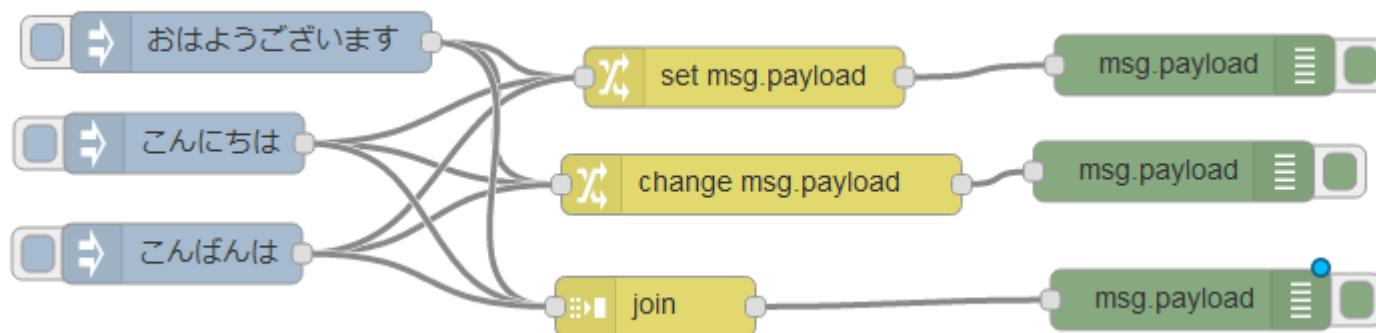


<https://github.com/WLO-RaspiClub/Node-RED/blob/master/nengo-service.json>



## ■Flowによるロジック

### ・メッセージ変更



**Edit change node**

Delete Cancel Done

▼ node properties

Name

Rules

Change ▼ msg.payload

Search for ▼ a\_z こんにちは

Replace with ▼ a\_z コン

**Edit join node**

Delete Cancel Done

▼ node properties

Mode

Combine each ▼ msg.payload

to create a String ▼

joined using ▼ a\_z ,

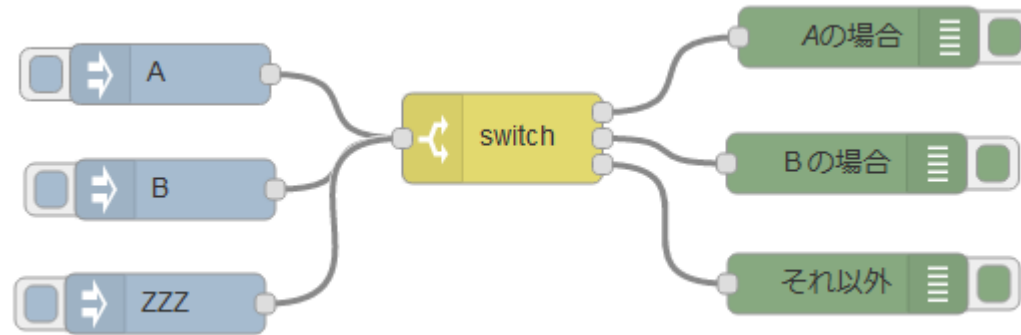
Send the message:

- After a number of message parts
- After a timeout following the first message
- After a message with the msg.complete property set

Name

## ■Flowによるロジック

### ・条件分岐



**Edit inject node** [Cancel] [Done]

**Payload**

**Topic**

**Repeat**

☐ Inject once at start?

**Name**

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

**Edit switch node** [Cancel] [Done]

**Name**

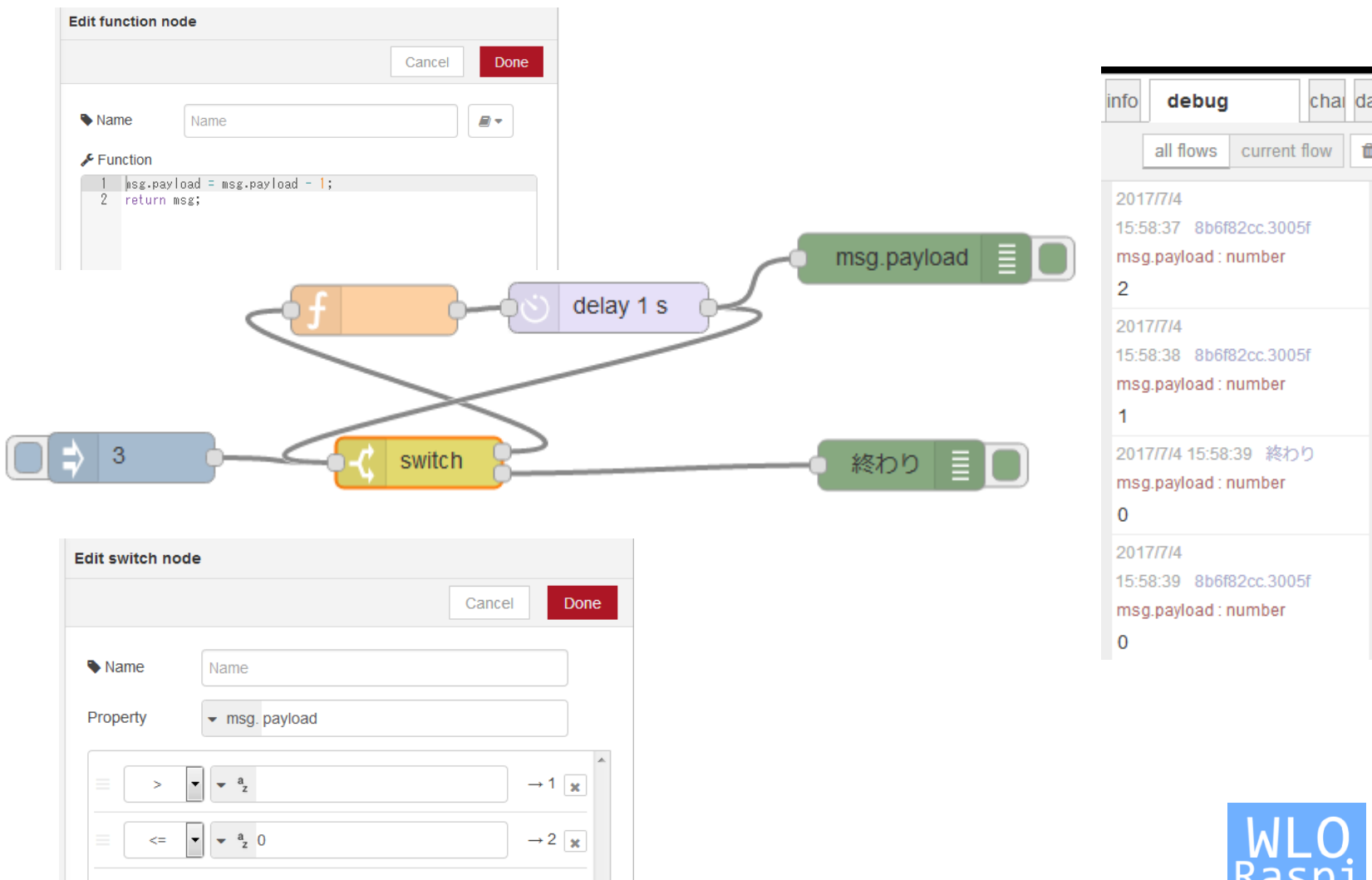
**Property**

==	<input type="text" value="A"/>	→ 1
==	<input type="text" value="B"/>	→ 2
otherwise		→ 3

checking all rules

## ■Flowによるロジック

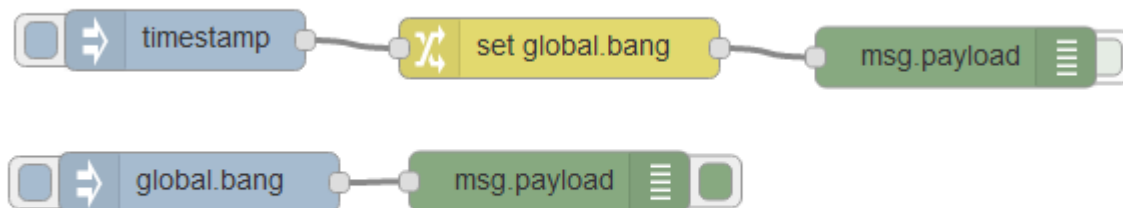
### ・ループ



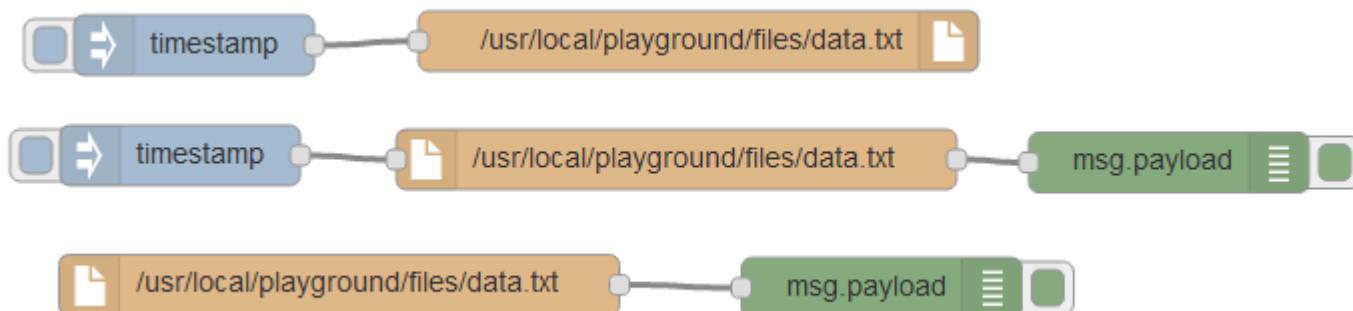
## ■Flowによるロジック

### ・永続化









Flow/Global変数



### ファイル











## ■さまざまなNode

▼ input	標準or カスタム	
 inject	標準	Flowエディタ操作（クリック）をトリガとしてFlowにメッセージを出力する。 定期送信機能（インターバル/時間指定/期間指定）も持つ。
 catch	標準	Tab内のJavascript例外を捕らえてFlowにメッセージを出力する。 処理異常時のFlow定義に用いる。
 status	標準	Tab内のNodeの状態変化を捕らえてFlowにメッセージを出力する。
 http	標準	インターネットからのhttpsのリクエスト(GET/POST/PUT/DELETE) を捕らえてFlowにメッセージを出力する。httpサービスのFlow定義に用いる。Flow中にhttp responseノードが1つ以上必要。
 websocket	標準	インターネットからのwebsocket (wss://) を捕らえてFlowにメッセージを出力する。WebsocketサービスのFlow定義に用いる。Flow中にwebsocket出力ノードが1つ以上必要。
▼ output		
 debug	標準	Flowからメッセージを受信してdebugサイドバーに出力する。サーバのコンソールに出力することもできる。
 http response	標準	Flowからメッセージを受信してhttpsクライアントにレスポンスを返す。http入力ノードとセットで用いる。
 websocket	標準	Flowからメッセージを受信してWebsocketクライアントにレスポンスを返す。websocket入力ノードとセットで用いる。


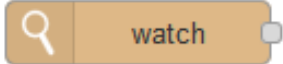
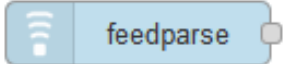

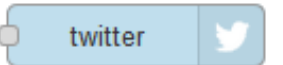
## ■さまざまなNode

▼ function	標準or カスタム	
 function	標準	Function Node : Javascriptで自由に処理を記載できるNode。出力を配列にすることにより、条件によって複数の出力を持つことができる。
 template	標準	メッセージのテンプレートを出力するNode。mustache/HTML/JSON/Markdown形式で文法チェックが可能。
 delay	標準	メッセージを遅延出力するNode。遅延量固定/ランダム/最大量指定/平滑化が可能。
 trigger	標準	2つのメッセージを入力情報/一定時間に応じて出力しわけするNode。GPIOのトグル出力などに用いる。
 comment	標準	Flowに注釈を記載するNode。処理には関係しない。
 http request	標準	外部Webサービスにリクエストを行うNode。GET/PUT/POST/PATCH/DELETEメソッドによる要求が可能で、ヘッダ/ボディのカスタマイズも可能。レスポンスを出力する。
 switch	標準	条件にしたがって出力を分岐するNode。Function Nodeでも記載可能だがより視覚的。
 change	標準	メッセージの作成/交換/削除を行うNode。Function Nodeでも記載可能だがより視覚的。
 range	標準	入力された数値メッセージのスケーリングを行うNode。特定の範囲を持つセンサ入力を正規化したり上限/下限を設定したりできる。
 csv	標準	CSVやTSVのように、区切り文字で区切られた文字列をJavascriptのオブジェクトに変換したり、その逆を行うNode。
 html	標準	HTMLを加工するNode。CSSセレクタを用いてページ情報を文字列だけにしたり整形したりできる。
 json	標準	JSON文字列をJavascriptのオブジェクトに変換したり、その逆を行うNode。
 xml	標準	xml文字列をxml2jsで解析してオブジェクトに変換したり、その逆を行うNode。

## ■さまざまなNode

▼ storage	標準or カスタム	
 tail	標準	サーバ上のファイルが追記されたときに追記内容をFlowに出力する。行ごとにメッセージを分けることも可能。
 file	標準	サーバ上のファイルの内容をFlowに出力する。ファイル名をFlowから入力して指定することもできる。行ごとにメッセージを分けることも可能。
 amazon s3	カスタム	Amazon S3の特定のBuckets/Folderに変化があればFlowに出力する。Amazon IAM クレデンシャルを登録する必要がある。
 sqlite	カスタム	サーバ上のSQLiteデータベースファイルを操作する。Flowからの入力でCREATE/INSERT/SELECT/UPDATE/DELETEのSQL文を入力でき、結果をFlowに出力する。
file 	標準	Flowからメッセージを入力して、サーバ上のファイルにメッセージを出力する。
 postgres	カスタム	PostgreSQL データベースに接続してSQLを送信する。
 amazon s3	カスタム	Amazon S3のオブジェクトへメッセージを出力する。Amazon IAM クレデンシャルを登録する必要がある。
 amazon s3	カスタム	Amazon S3のオブジェクトを読み込みFlowへ出力する。Amazon IAM クレデンシャルを登録する必要がある。
▼ function		
 rbe	標準	連続して入力された数値メッセージの値が変化したときのみ出力するNode。出力条件に範囲指定も可能。数値が変化したときのみ実施するFlowを作成する時などに用いる。

## ■さまざまなNode

▼ analysis	標準or カスタム	
 sentiment	標準	入力された文字列を英語の文字列とみなして、AFINN-111を用いて感情判定しFlowに出力する。英語のみ有効。
▼ advanced		
 watch	標準	サーバ上の指定したフォルダ内のファイルが変更される際にメッセージをFlowに出力する。
 feedparse	標準	指定したURLのRSS/atomフィードを定期的に読み込み、新しいエントリが登録されたらFlowに出力する。
▼ social		
 twitter	標準	Twitter APIを用いて、公開ツイート/フォロアツイート/メンション/DMを取得しFlowに内容を出すNode。Twitter APIのクレデンシャルが必要。
 twitter	標準	Twitter APIを用いて、公開ツイート/フォロアツイート/メンション/DMを送信するNode。Twitter APIのクレデンシャルが必要。



## ■ CustomNodeの導入

The screenshot displays the Node-RED web interface. A 'User Settings' dialog box is open, showing the 'Nodes' tab. The 'Palette' section lists three custom nodes:

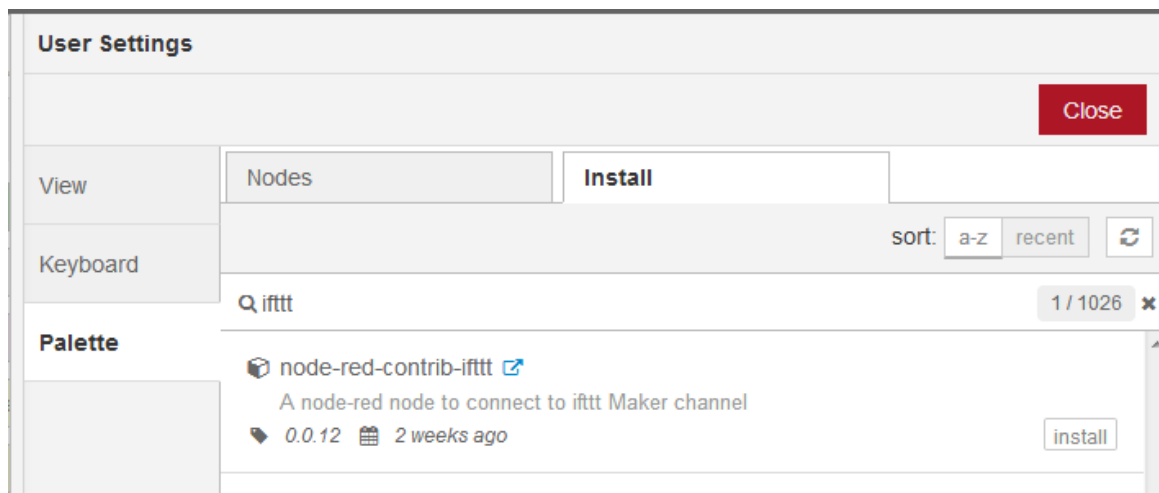
- node-red-contrib-quotes**: Get a random quote from different sources. Version 0.2.1, installed 2 months ago. [install]
- node-red-contrib-vimeo-random**: Random call between outputs. Version 0.0.4, installed 4 months ago. [install]
- node-red-node-random**: A Node-RED node that when triggered generates a random number between two values. Version 0.0.8, installed 7 months ago. [installed]

The right panel shows the 'Flow' information for 'Flow 3' with ID 'fab2a41d.b06d18' and status 'Enabled'. At the bottom, a message states: 'Pressing **enter** will edit the first node in the current selection'.

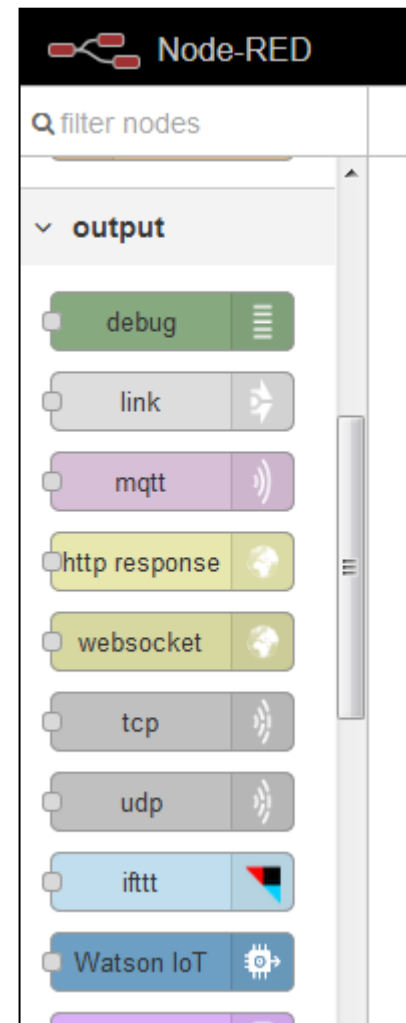
- Custom Nodeの導入

<https://flows.nodered.org/node/node-red-contrib-ifttt>

ハンバーガメニュー⇒Manage Palette、Installタブで「ifttt」で検索

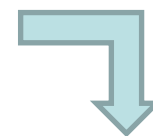
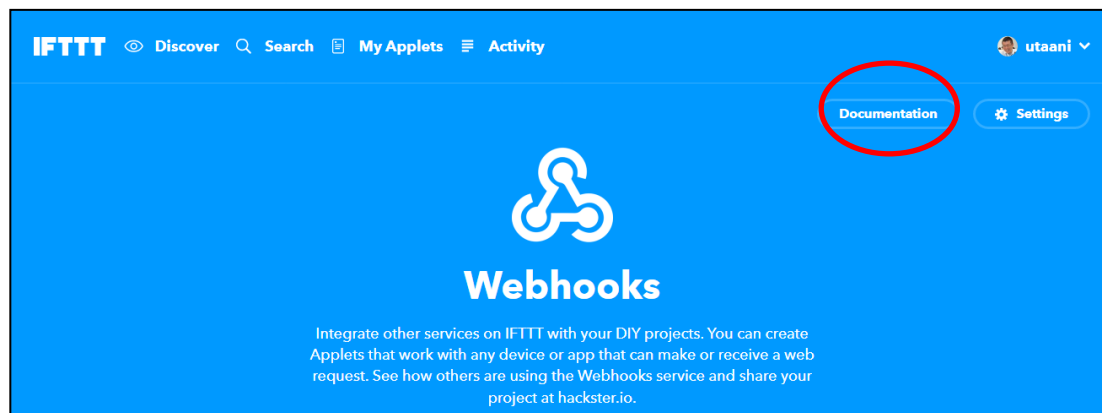


インストールできると、Output Nodeの欄に「ifttt」Nodeが表示される

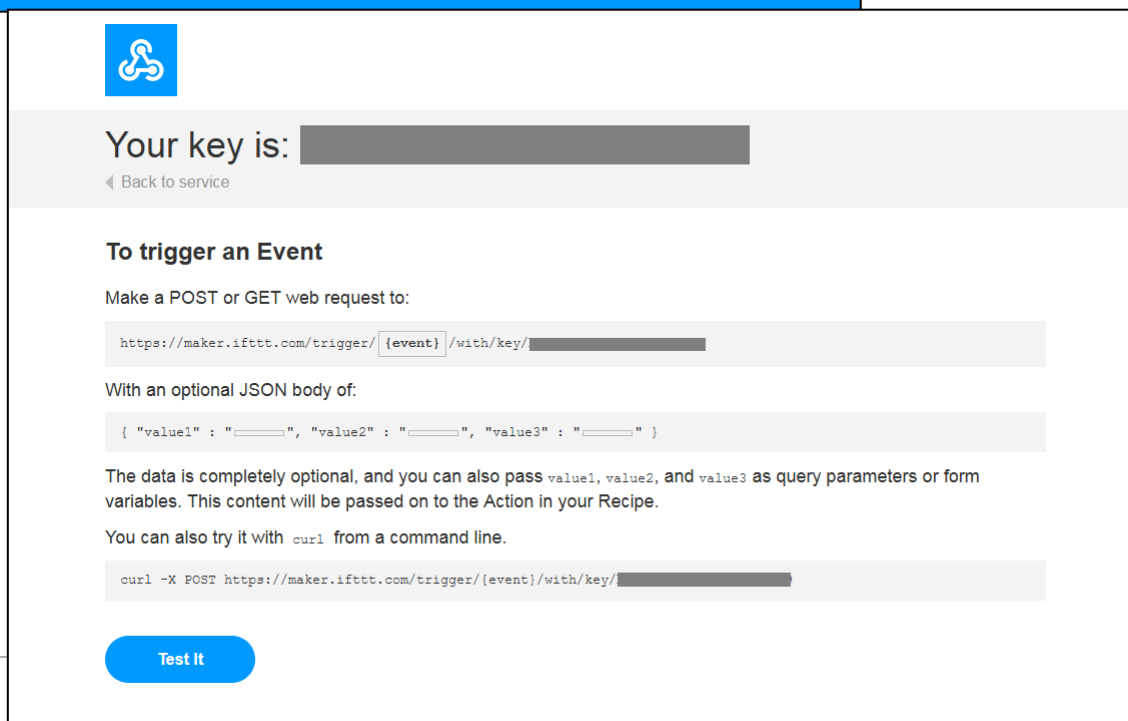


## ・IFTTTのWebhooksの設定

[https://ifttt.com/maker\\_webhooks](https://ifttt.com/maker_webhooks)

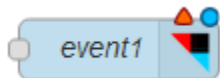


Documentation



APIキーを  
コピーしておく

## ・IFTTT Nodeの設定




Nodeを配置してダブルクリック

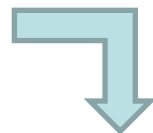
Edit ifttt out node

Delete Cancel Done

▼ node properties

🔑 Key Add new ifttt-key... 

⚡ Event Name event1

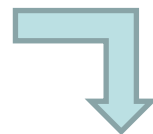


ifftt out > Add new ifttt-key config node

Cancel Add

🔑 Key

APIキーを入力して  
Add



Edit ifttt out node

Delete Cancel Done

▼ node properties

🔑 Key

⚡ Event Name event1

Event Nameを  
指定して  
Done

- ・IFTTT Appletの設定

## ■ FunctionNode版Hello World



**Edit function node**

Delete

Cancel

Done

▼ node properties

Name

Function

```
1 msg.payload = "hello world";
2 return msg;
```

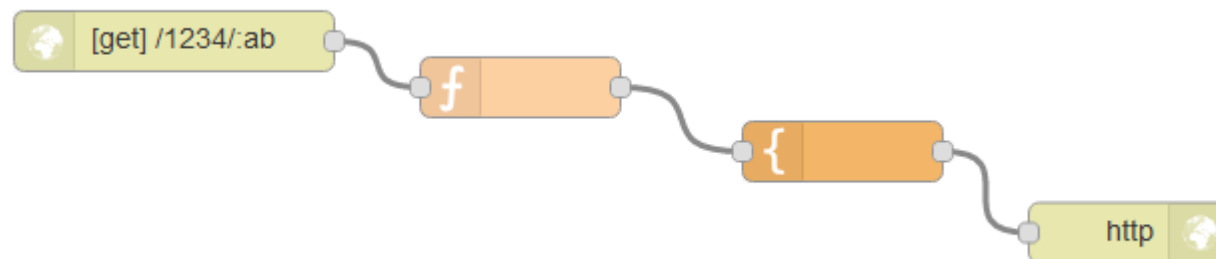
Outputs

1

See the Info tab for help writing functions.

> port labels

## ■FunctionNodeのロジックを記載する



**Edit function node**

Cancel

Done

Name

Name

Function

```
1 if(msg.req.params.ab === 'hello') {
2   msg.contents = "こんにちは";
3 }
4 else {
5   msg.contents = "いらっしゃいませ";
6 }
7 return msg;
```

## ・Raspberry Piのシャットダウン



Edit exec node

Delete Cancel Done

node properties

Command

+ Append ☐ msg.payload

Output

☐ Use old style output (compatibility mode)

Timeout  seconds

Name



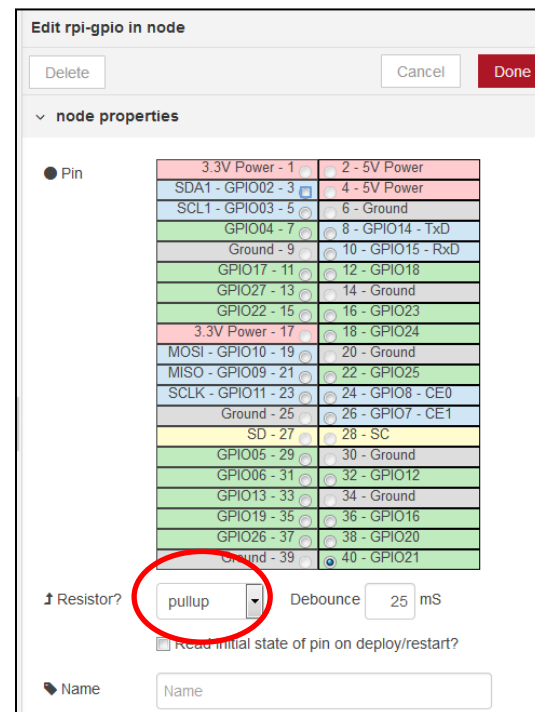
## ・GPIOのスイッチインプット

スイッチを39pin/40pinに接続する



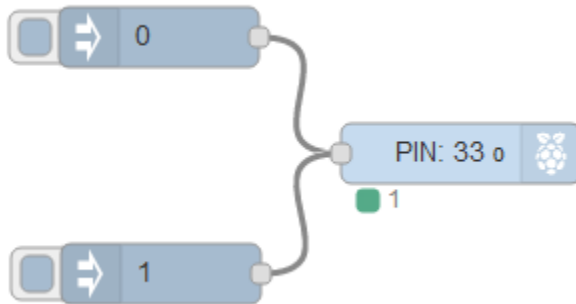
GNDに接続する場合は、Resistor(抵抗器)をPullupに設定しておく。

⇒ 普段は1が出力、押されたら0が出力



## ・GPIOからLED出力

LEDを33pin/34pinに接続する。  
(34pin側にLEDの足短いほうを接続)



Numberの0で消灯、1で点灯

Edit rpi-gpio out node

Delete Cancel Done

▼ node properties

● Pin

3.3V Power - 1	2 - 5V Power
SDA1 - GPIO02 - 3	4 - 5V Power
SCL1 - GPIO03 - 5	6 - Ground
GPIO04 - 7	8 - GPIO14 - TxD
Ground - 9	10 - GPIO15 - RxD
GPIO17 - 11	12 - GPIO18
GPIO27 - 13	14 - Ground
GPIO22 - 15	16 - GPIO23
3.3V Power - 17	18 - GPIO24
MOSI - GPIO10 - 19	20 - Ground
MISO - GPIO09 - 21	22 - GPIO25
SCLK - GPIO11 - 23	24 - GPIO8 - CE0
Ground - 25	26 - GPIO7 - CE1
SD - 27	28 - SC
GPIO05 - 29	30 - Ground
GPIO06 - 31	32 - GPIO12
GPIO13 - 33	34 - Ground
GPIO19 - 35	36 - GPIO16
GPIO26 - 37	38 - GPIO20
Ground - 39	40 - GPIO21

Type: Digital output

☒ Initialise pin state?

initial level of pin - low (0)

Name:

Node-RED <a href="http://nodered.org/">http://nodered.org/</a>	Node-REDの配布/解説サイト
Node-RED documentation <a href="http://nodered.org/docs/">http://nodered.org/docs/</a>	Node-REDドキュメント
Node-RED Library <a href="http://flows.nodered.org/">http://flows.nodered.org/</a>	Node-REDのサンプルFlowやカスタムNode配布サイト
Github: node-red/node-red <a href="https://github.com/node-red/node-red">https://github.com/node-red/node-red</a>	Node-RED Githubリポジトリ
GoogleGroup:Node-RED <a href="https://groups.google.com/forum/#!forum/node-red">https://groups.google.com/forum/#!forum/node-red</a>	Node-RED 公式MailingList/Googlegroup
Node-RED Programming Guide <a href="http://noderedguide.com/">http://noderedguide.com/</a>	sence technic社による解説ページ
Qiita: tag:node-red <a href="http://qiita.com/tags/node-red">http://qiita.com/tags/node-red</a>	Qiitaは、プログラミングに関する知識を記録・共有するためのサービスです。
Node-RED Advent Calendar 2015 <a href="http://qiita.com/advent-calendar/2015/nodered">http://qiita.com/advent-calendar/2015/nodered</a>	2015年末に開催された Node-REDの情報を集めたアドベントカレンダー
Node-RED Advent Calendar 2016 <a href="http://qiita.com/advent-calendar/2016/nodered">http://qiita.com/advent-calendar/2016/nodered</a>	2016年末に開催された Node-REDの情報を集めたアドベントカレンダー
Node-RED User Group Japan <a href="https://www.facebook.com/groups/noderedjp/">https://www.facebook.com/groups/noderedjp/</a>	日本のユーザーグループ(登録制)
IBM developerWorks <a href="http://www.ibm.com/developerworks/jp/views/cloud/libraryview.jsp?sort_by=&amp;search_by=node-red">http://www.ibm.com/developerworks/jp/views/cloud/libraryview.jsp?sort_by=&amp;search_by=node-red</a>	IBMの技術文書