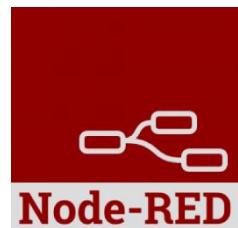
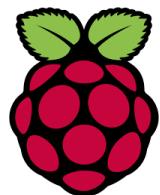


WLOラズパイ俱楽部+a

ラズパイ + Node-REDで IoTにチャレンジ 2019

2019年10月10日



<http://git.io/wlopi>

WLO
Raspi
Club+a

<https://github.com/WLO-RaspiClub/>

<https://git.io/wlopi>

過去23回の
「WLOラズパイ俱楽部」の
資料、スクリプトが
格納されています。

今後のイベントでも
ここに資料を格納するので
予習・復習にご活用ください。

The screenshot shows the GitHub repository page for 'WLO-RaspiClub'. The repository has 25 repositories, 5 people, and no teams. It features a search bar and filters for type and language. Four recent projects are listed:

- 20191010_Node-RED**: A project for Node-RED, dated October 10, 2019. It has 0 stars, 0 issues, 0 pull requests, and was updated 3 days ago.
- 20190822_VoiceCommandWithJulius**: A project for VoiceCommandWithJulius, dated August 22, 2019. It has 0 stars, 0 issues, 0 pull requests, and was updated on 22 Aug.
- 20190711_RaspberryPiInstallSupport**: A project for RaspberryPiInstallSupport, dated July 11, 2019. It has 0 stars, 0 issues, 0 pull requests, and was updated on 12 Jul.
- 20190606_HeadlessWithCloud**: A project for HeadlessWithCloud, dated June 6, 2019. It has 0 stars, 0 issues, 0 pull requests, and was updated on 6 Jun.



最近のRaspberry Piについて

- ・前回(2019/8/22)以降の新情報

Raspbian buster 2019-09-26リリース

<https://www.raspberrypi.org/downloads/raspbian/>

Raspbian Buster with desktop and recommended software
Image with desktop and recommended software based on Debian Buster

Version: September 2019
Release date: 2019-09-26
Kernel version: 4.19
Size: 1945 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256:
549da0fa09ed52a8d7c2d66cb06afac9fe856638b06d8f23df4e6b72e67ed4cea

Raspbian Buster with desktop
Image with desktop based on Debian Buster

Version: September 2019
Release date: 2019-09-26
Kernel version: 4.19
Size: 1149 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256:
2c4067d59acf891b7aa1683cb1918da78d76d2552c02749148d175fa7f766842

Raspbian Buster Lite
Minimal image based on Debian Buster

Version: September 2019
Release date: 2019-09-26
Kernel version: 4.19
Size: 426 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256:
a50237c2f718bd8d806b96df5b9d2174ce8b789eda1f03434ed2213bbca6c6ff

更新情報 http://downloads.raspberrypi.org/raspbian/release_notes.txt
(日本語訳) <https://qiita.com/utaani/items/85f6e6515ed576cadb9a>

今回のゴール

■ Raspberry Pi + Node-REDでIoTを体験

- ・RaspbianにNode-REDの最新バージョンを導入
- ・最初のFlow - Hello World
- ・セキュリティ設定
- ・さまざまなFlowを書いてみる
- ・カスタムノード導入
- ・GPIO活用

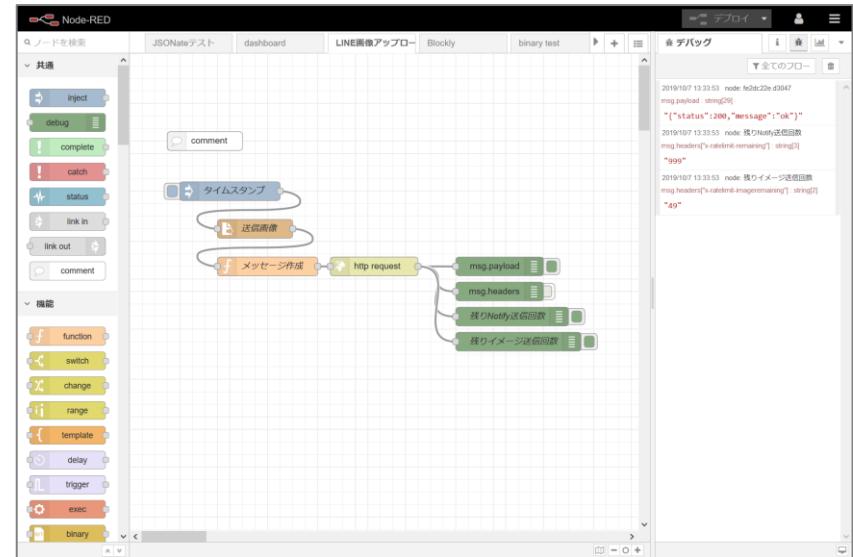
Node-REDとは：Webベースの統合開発環境

5

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

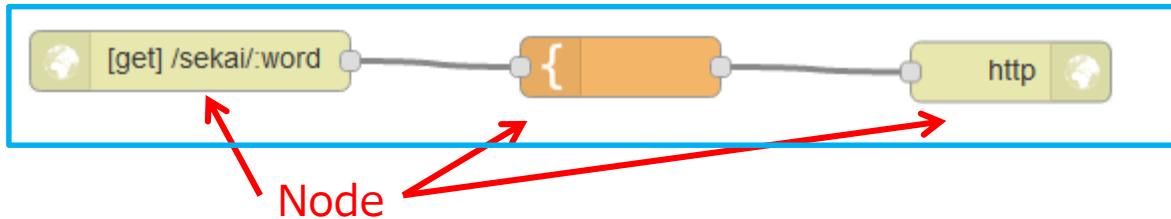
ライセンスは「Apache 2.0 License」

- 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。



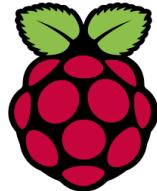
Flow

- ・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。

どの環境でも同じように実装できる

組み込み環境



PC/サーバ



パブリッククラウド



Google Cloud Platform



IBM Bluemix



導入方法 <https://nodered.org/docs/getting-started/installation>

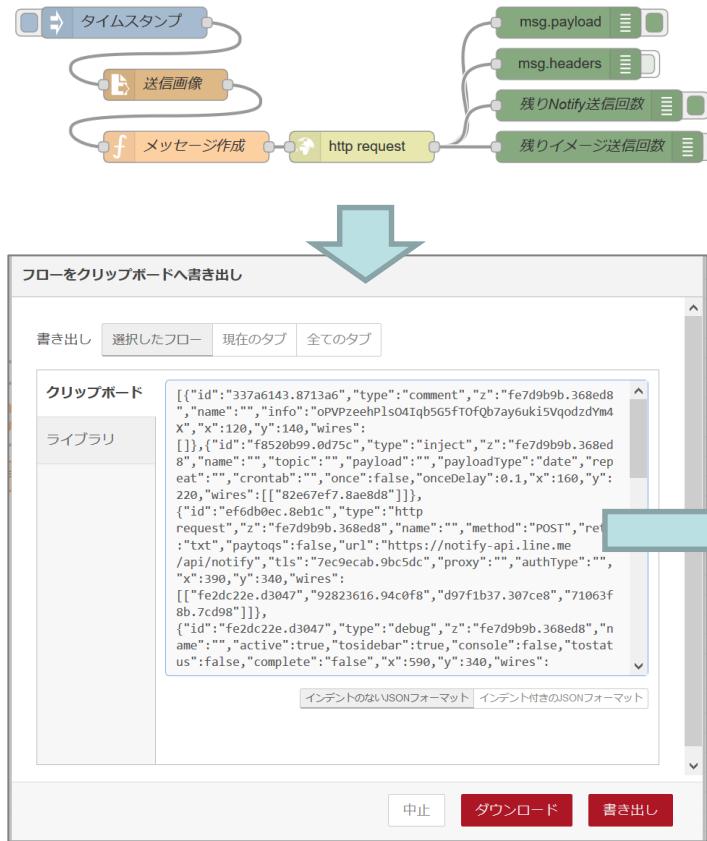
<http://git.io/wlopi>

Node-REDの特徴

7

• 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



Node-REDの特徴

・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。

The diagram illustrates the process of adding custom nodes from the Node-RED Library to the Node-RED palette:

- Node-RED Library:** Shows the main interface with sections for Recent nodes, Recent flows, and Recent collections.
- node-red-contrib-amazon-echo 0.1.9:** A detailed view of a specific library entry, showing its description, version, ratings, and installation instructions via npm.
- Manage Palette - Nodes:** A screenshot of the Node-RED interface showing the 'Nodes' tab with the newly installed 'amazon-echo-hub' and 'amazon-echo-device' nodes listed.
- Flow Editor - Palette:** A screenshot of the Node-RED Flow Editor showing the palette with the newly added 'amazon echo hub' and 'amazon echo device' nodes.

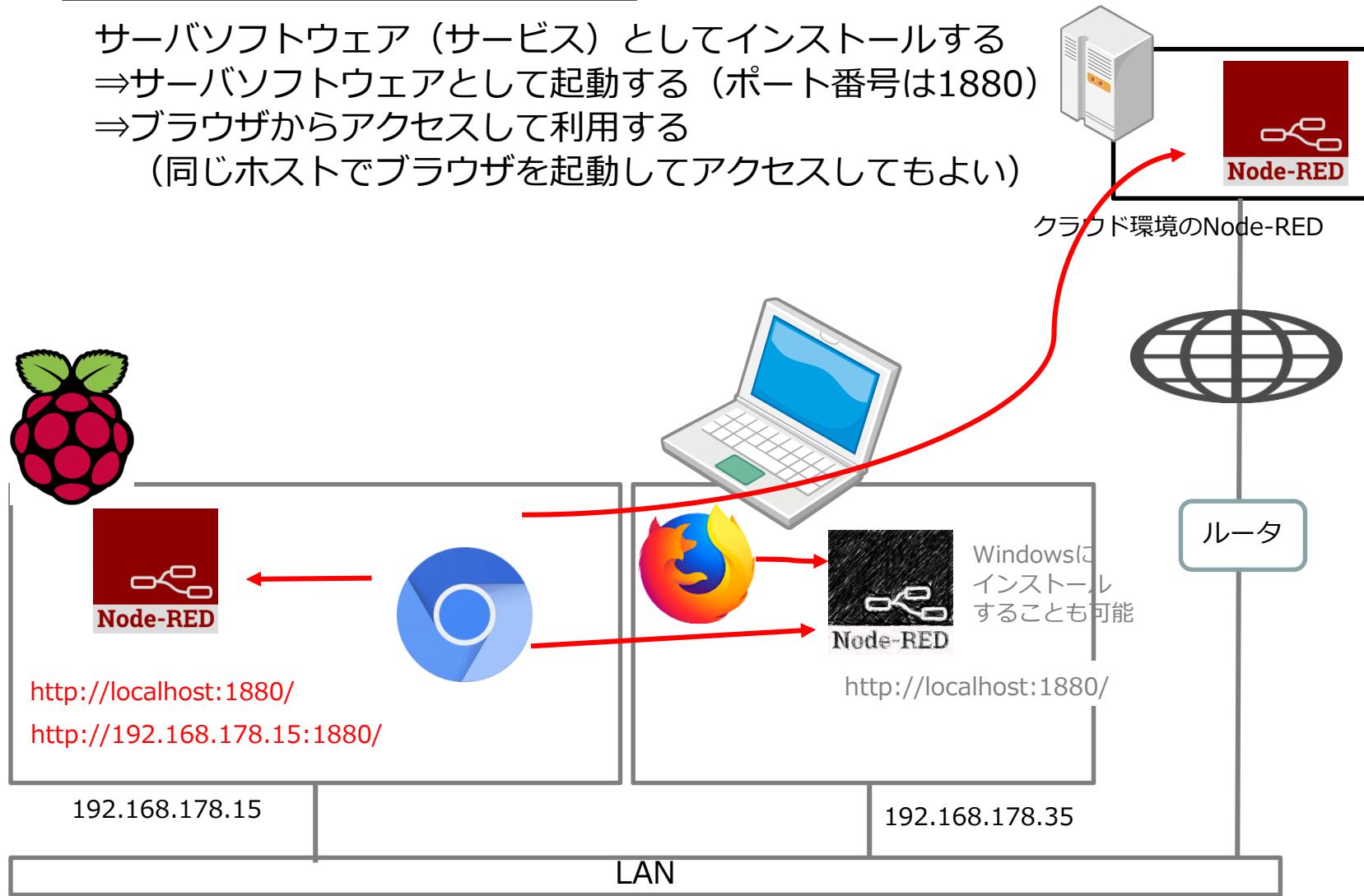
Arrows indicate the flow from the library search results to the detailed node page, then to the Manage Palette, and finally to the Flow Editor palette.

<http://flows.nodered.org/>

WLO Raspi Club

■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)



Node-REDのインストール

■公式インストールテキスト

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

The screenshot shows the "Running Node-RED locally" section of the Node-RED documentation. It includes a sidebar with prerequisites like npm, Docker, and snap installations, and sections for installing with npm, Docker, Snap, and command-line usage. It also covers passing arguments to the underlying Node.js process and upgrading Node-RED. A "Next steps" section is at the bottom.

Prerequisites

- Installing with npm
- Installing with docker
- Installing with snap
- Running
- Command-line Usage
- Passing arguments to the underlying Node.js process
- Upgrading Node-RED
- Next steps

Running Node-RED locally

Prerequisites

To install Node-RED locally you will need a [supported version of Node.js](#).

If you are on a Raspberry Pi or any Debian-based operating system, including Ubuntu and Diet-Pi, you can use the Pi install script available [here](#).

If you are on an RPM-based operating system, including RedHat, Fedora and CentOS, you can use the RPM install script available [here](#).

Installing with npm

To install Node-RED you can use the `npm` command that comes with node.js:

```
sudo npm install -g --unsafe-perm node-red
```

If you are using Windows, do not start the command with `sudo`. More information on Windows installation can be found [here](#).

This command will install Node-RED as a global module along with its dependencies.

<https://nodered.jp/docs/getting-started/local>

The screenshot shows the "ローカルでNode-REDを実行する" (Run Node-RED locally) section of the Node-RED User Group Japan documentation. It includes a sidebar with prerequisites like npm, Docker, Snap, and command-line usage, and sections for installing with npm, Docker, Snap, and command-line usage. It also covers upgrading Node-RED. A "Next steps" section is at the bottom.

必須条件

- npmによってインストールする
- Dockerでインストールする
- Snapでインストールする
- 実行する
- コマンドラインの使い方
- 基盤であるNode.jsプロセスに引数を渡す
- Node-REDをアップグレードする
- 次のステップ

ローカルでNode-REDを実行する

ローカル環境にNode-REDをインストールするためにはサポートされているNode.jsのバージョンが必要です。

Raspberry PiまたはUbuntuまたはDiet-Piを含むDebianベースのオペレーティングシステムの場合、こちらのRaspberry Piのインストールスクリプトを利用できます。

RedHat、FedoraおよびCentOSを含むRPMベースのオペレーティングシステムを利用している場合、こちらで入手できるRPMのインストールスクリプトを利用することができます。

npmによってインストールする

Node-REDをインストールするため、Node.jsに同梱の `npm` コマンドを利用できます:

```
sudo npm install -g --unsafe-perm node-red
```

Windowsを利用している場合、このコマンドを `sudo` から始めません。Windowsでのインストールに関する追加情報はこちで確認できます。

このコマンドは、依存関係とともにNode-REDをグローバルモジュールとしてインストールします。

※Node-RED UG Japan有志による翻訳

Raspberry PiにおけるNode-REDの活用について

<https://qiita.com/utaani/items/7155c62d6c5e96822afb>

The screenshot shows a Qiita article page. At the top, there's a navigation bar with 'Qiita' and search fields. Below it, the author's profile (@utaani) and the date (2019年10月07日) are shown, along with 7699 views. A red banner indicates it's part of the 'Node-RED Advent Calendar 2018 | 4日目'. The main title 'Raspberry PiにおけるNode-REDの活用について' is displayed prominently. Below the title are tags: 'RaspberryPi' and 'node-red'. The article content starts with a paragraph about the use of Node-RED on Raspberry Pi. It then discusses the purpose of the article, mentioning the Node-RED Advent Calendar 2018, and concludes with a note about the current Node-RED version (1.0.1). A section at the bottom is titled 'Raspberry Piって何?' and provides a brief introduction to what a Raspberry Pi is.



Raspberry Piで
Node-REDを活用する
ポイントがまとめてあります。
(バージョン更新時に
メンテナンスしています)

RaspbianへNode-REDをインストール

Raspberry PiにおけるNode-REDの活用について

<https://qiita.com/utaani/items/7155c62d6c5e96822afb>

RaspbianへのNode-REDのインストール

「Raspbian Buster with desktop and recommended software」を導入された場合、Node-REDは最初から導入されています。メニューの「プログラミング」にNode-REDがありますので、ここから実行できます。それ以外のイメージから導入された場合は、まずはインストールが必要です。

「Raspbian Buster with desktop」で導入した場合、Node-REDのインストールには、PIXEL Desktopのメニューにある「Recommended Software」を使っても導入できますが、時期によっては古いバージョンを導入することになり、導入後に別途更新作業が必要となり二度手間です。

おすすめのNode-RED導入方法としては、Node-REDのリポジトリでメンテナンスされている[update-nodejs-and-nodered](#)スクリプトを最初から使いましょう。(まったく導入されていない状態から使えるスクリプトです)
公式ページ内の[Running on Raspberry Pi](#)で紹介されている

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

でインストールできますが、URLが長いので覚えにくいのが難点です。githubで提供されている短縮URLサービスを使って <https://git.io/noderedlinux> という短縮URLを作りましたので、コンソールから

```
wget https://git.io/noderedlinux
less noderedlinux
(スクリプトを確認※)
bash noderedlinux
```

で導入可能です。noderedlinux(update-nodejs-and-nodered)は内部でsudoしていますので、piユーザがsudoで管理者権限になれないように設定している場合以外は、piユーザで実行します。(rootユーザで実行するとスクリプトが実行されません)

インストールスクリプトの取得と実行

```
$ wget https://git.io/noderedlinux ↵
$ bash noderedlinux ↵
```

※一度再度実行する場合は
前回ダウンロードしたスクリプトを削除してから
再実施する

```
$ rm noderedlinux ↵
$ wget https://git.io/noderedlinux ↵
$ bash noderedlinux ↵
```

インストールスクリプトからの確認

This script will remove versions of Node.js prior to version 7.x, and Node-RED and if necessary replace them with Node.js 10.x LTS (dubnium) and the latest Node-RED from Npm.

It also moves any Node-RED nodes that are globally installed into your user `~/.node-red/node_modules` directory, and adds them to your `package.json`, so that you can manage them with the palette manager.

It also tries to run '`npm rebuild`' to refresh any extra nodes you have installed that may have a native binary component. While this normally works ok, you need to check that it succeeds for your combination of installed nodes.

To do all this it runs commands as root - please satisfy yourself that this will not damage your Pi, or otherwise compromise your configuration.
If in doubt please backup your SD card first.

Are you really sure you want to do this ? [y/N] ? **Y**

y ↲ と入力

Would you like to install the Pi-specific nodes ? [y/N] ? **Y**

y ↲ と入力

参考日本語訳:

このスクリプトはバージョン7.x以前のNode.jsを削除し、必要に応じて、Node.js 10.x LTSとNode-REDをNpmで最新版に置き換えます。また、ユーザによりグローバルにインストールされているNode-REDノードを`.node-red/node_modules`ディレクトリに移動し、`package.json`を追加してパレットマネージャで管理できるようにします。さらに、「`npm rebuild`」を実行してインストール済みの外部ノードのネイティブバイナリーコンポーネントを更新します。普通は問題なく動作しますが、導入されたノードの相性によっては、きちんと完了したか確認する必要があるかもしれません。
全てのコマンドはrootで実行されますので、自分自身あなたのラズパイにダメージを与えないようにするか、設定を変更してください。怪しい場合は、まずSDカードのバックアップを取ってください。

本当にこれを実施しても大丈夫ですか? [y/N] ?

ラズパイ固有のノードをインストールしてもいいですか? [y/N]

インストールスクリプト実行:だいたい15分くらいかかります

```
pi@raspberrypi: ~
▼ ▲ X

ファイル(F) 編集(E) タブ(T) ヘルプ(H)

Running Node-RED install for user pi at /home/pi on raspbian

This can take 20-30 minutes on the slower Pi versions - please wait.

Stop Node-RED          ✓
Remove old version of Node-RED   ✓
Remove old version of Node.js    ✓
Install Node.js LTS          ✓ Node v10.16.3 Npm 6.11.3
Clean npm cache            ✓
Install Node-RED core        ✓ 1.0.0
Move global nodes to local   ✓
Install extra Pi nodes       ✓
Npm rebuild existing nodes   -
Add shortcut commands        ✓
Update systemd script        ✓

Any errors will be logged to /var/log/nodered-install.log
All done.

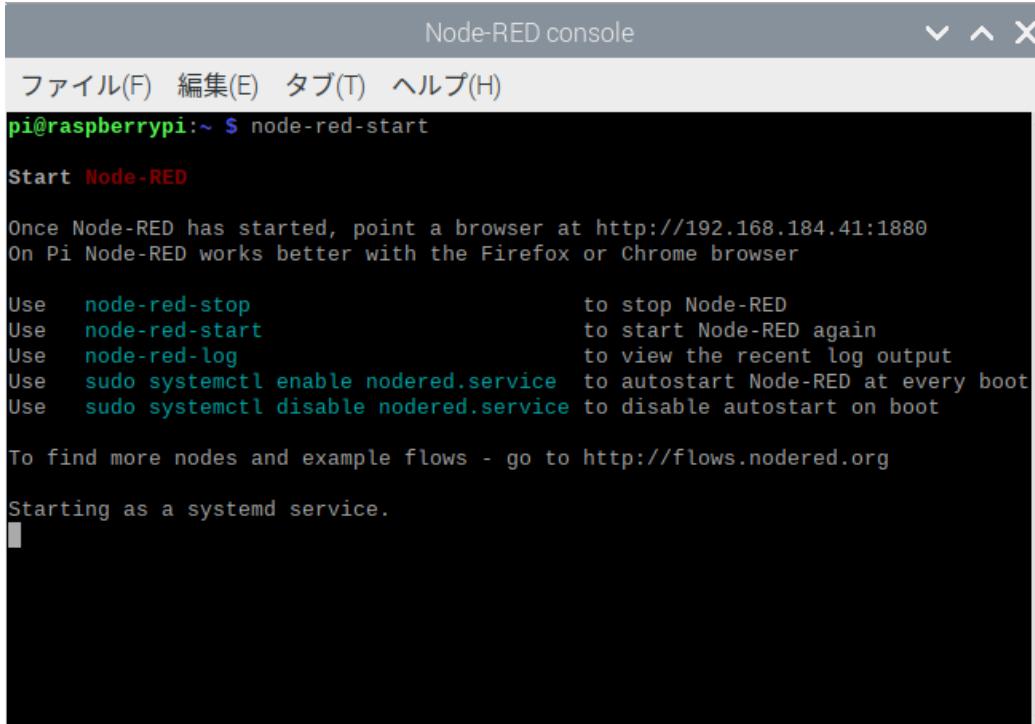
You can now start Node-RED with the command node-red-start
or using the icon under Menu / Programming / Node-RED
Then point your browser to localhost:1880 or http://{your_pi_ip-address}:1880

Started 2019年 10月 2日 水曜日 09:08:28 JST - Finished 2019年 10月 2日 水曜日 09:22:15 JST

pi@raspberrypi:~ $
```

■Node-RED最初の実行

```
$ node-red-start ↵
```



The screenshot shows a terminal window titled "Node-RED console". The window has a menu bar with "ファイル(F)", "編集(E)", "タブ(T)", and "ヘルプ(H)". The main area displays the output of the command \$ node-red-start. The output includes instructions for starting and stopping the service, information about autostarting, and a link to the Node-RED documentation. It ends with the message "Starting as a systemd service." followed by a progress bar.

```
Node-RED console
▼ ▲ X

ファイル(F) 編集(E) タブ(T) ヘルプ(H)
pi@raspberrypi:~ $ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.184.41:1880
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

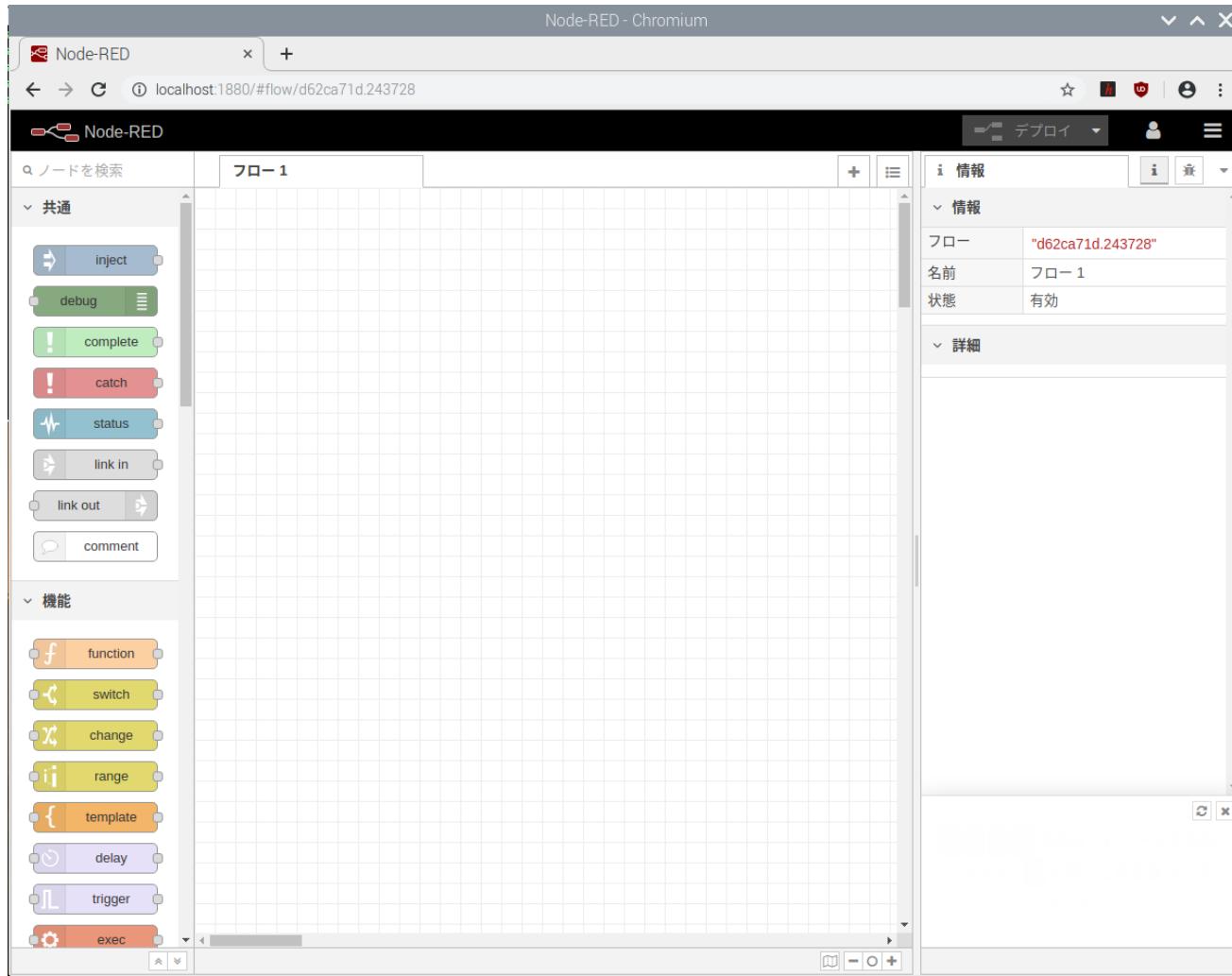
To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
█
```

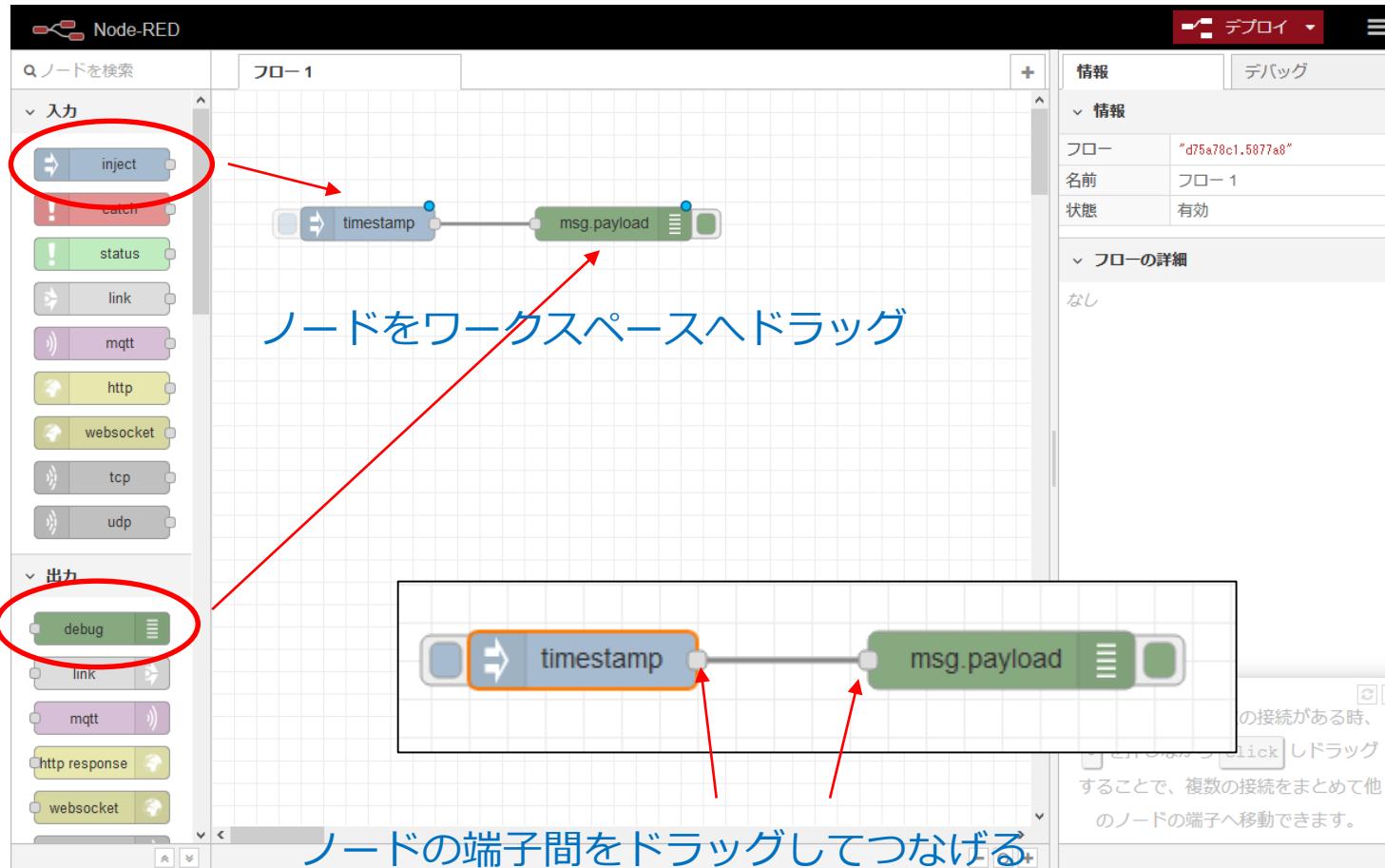
※ここまで動作したらサービスとして起動しているので、
Ctrl-Cで止めてもNode-REDは稼働しつづける

■Node-RED最初の実行

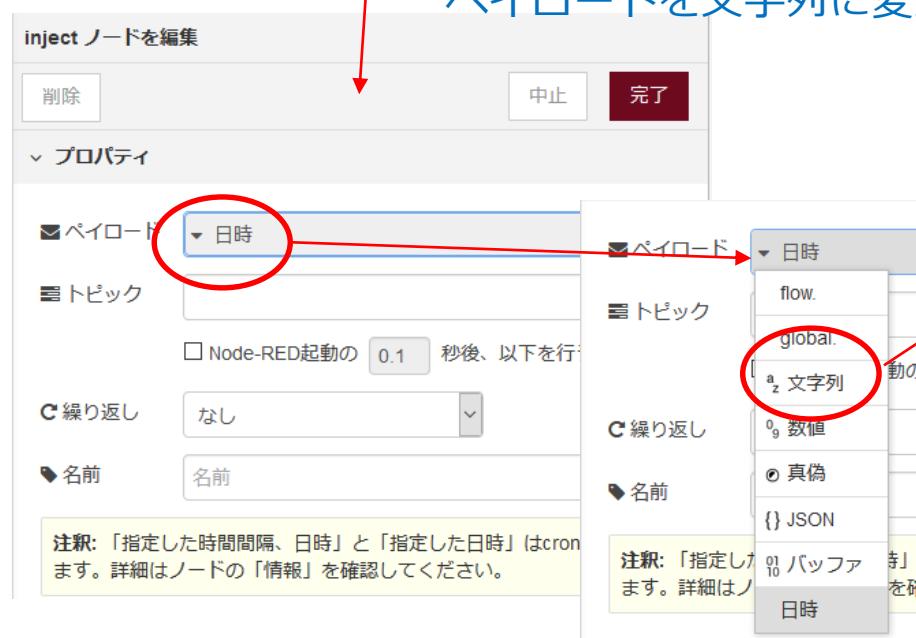
ブラウザで localhost:1880 にアクセスしてFlowEditorが起動したらインストール成功



■最初のフロー：Hello World

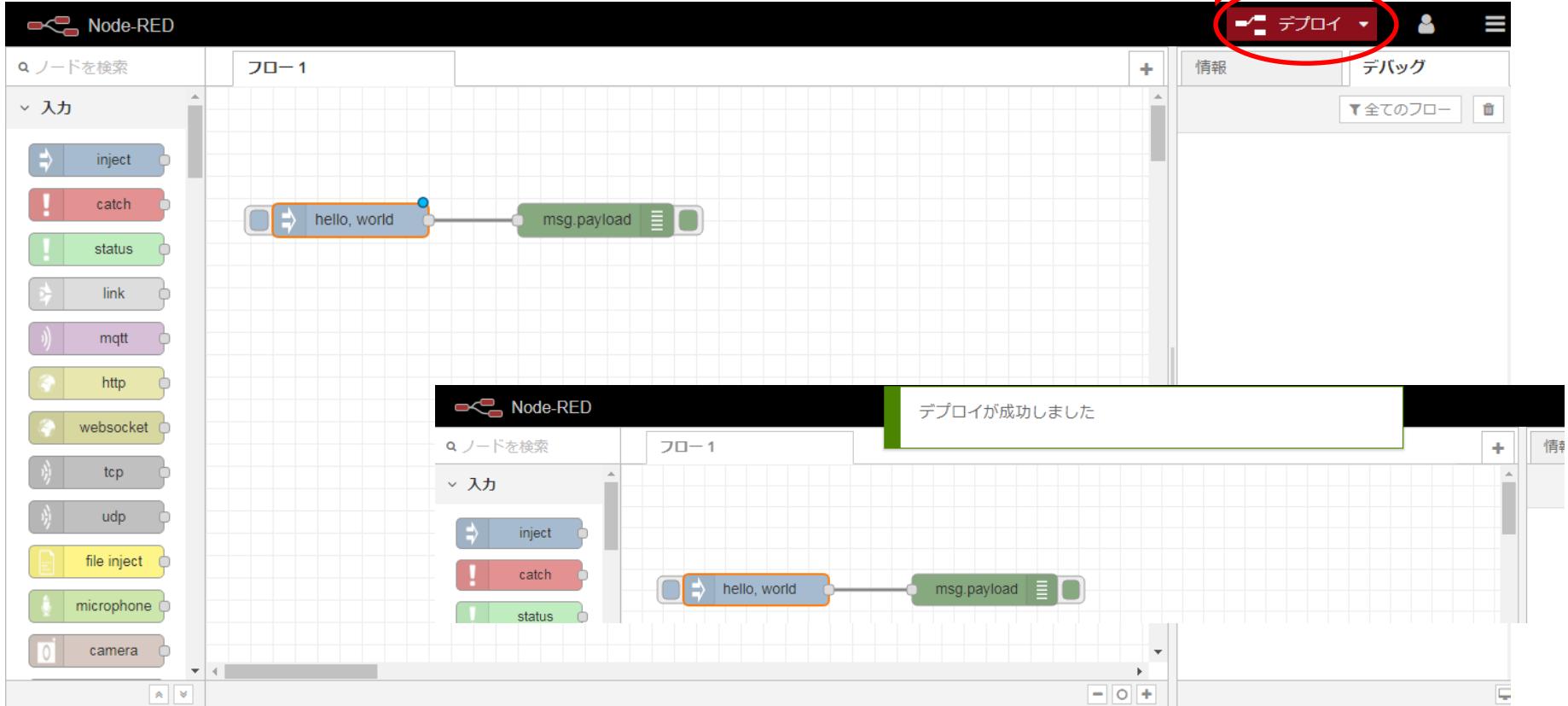


■最初のフロー：Hello World



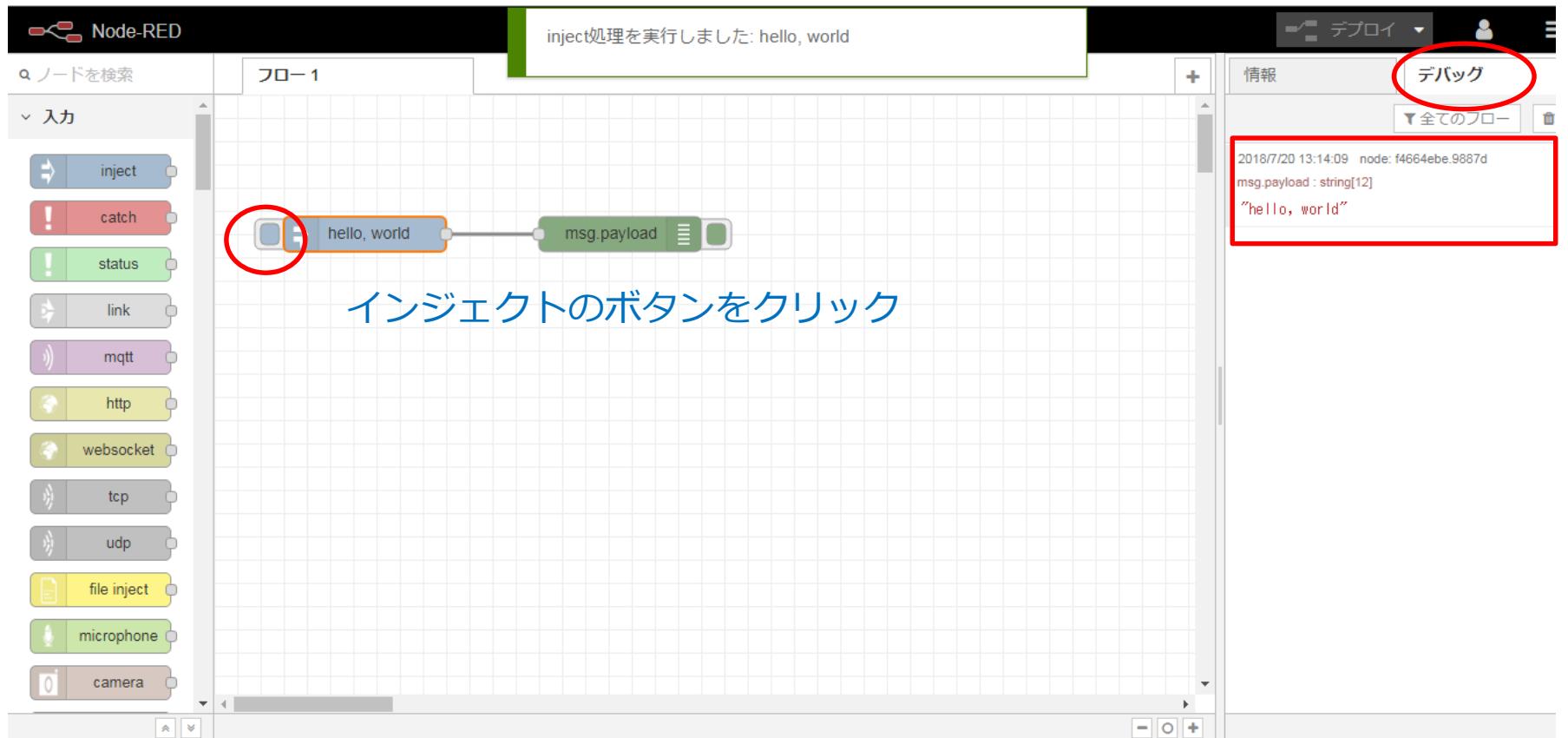
■最初のフロー：Hello World

デプロイをクリック



■最初のフロー：Hello Worldの実行

デバッグタブに文字列が表示される



■デフォルトの設定では危険

標準インストールしただけでは、IPアドレスがわかると
だれでもFlow Editorにアクセスできます。

→Raspbianではpiユーザが管理者権限をもちますので、
FlowEditorからexecノードを使われてしまうと
管理者権限(root)でRaspberry Piの全操作が可能になってしまいます。

■最低限、FlowEditorのログイン画面を設定する



初回アクセス時に
←のような
ログイン画面を
表示して
ログインしないと
FlowEditorを
使えないように
設定することが可能です

Node-REDをセキュアにする

■スクリプトによる設定

<https://qiita.com/utaani/items/7a2a40a4547e4ba7d85c>

The screenshot shows a Qiita article page. At the top, there's a green header bar with the Qiita logo, a search bar, and navigation links for 'ホーム' and 'コミュニティ'. Below the header, the article title is displayed: 'Raspberry Pi上のNode-REDにパスワードを設定するスクリプト'. To the left of the main content area, there's a sidebar with social sharing icons for GitHub, LinkedIn, and Facebook, along with a '2' indicating two likes. The main content area includes a brief introduction, a code snippet titled 'Raspberry PiにおけるNode-REDで、セキュリティ設定を実施する', and two paragraphs of explanatory text.

Raspberry PiにおけるNode-REDで、セキュリティ設定を実施する

Raspberry PiにおけるNode-REDの活用について 記載したように、RaspbianでNode-REDを動作させるとpiユーザで実行されるため、Flow Editorからexecノードを使われてしまうと管理者権限でRaspberry Piの全操作が可能になってしまいます。

まったくインターネットからアクセスできないRaspberry Piでも、最低でもログイン画面設定は実施する必要がありますが、settings.jsの編集が必要なため、初心者には難しいという話がありました。

そこで、settings.jsを編集しパスワードを設定する「nrpiadminpass.sh」というスクリプトを作成しました。

nrpiadminpass.sh

Node-REDをセキュアにする

■スクリプトによる設定

<https://qiita.com/utaani/items/7a2a40a4547e4ba7d85c>

スクリプト導入方法

```
wget https://gist.githubusercontent.com/utaani/2243b7501a2cd52d93cb85c91f173e8/raw,
```

なお、URLが長いので、githubで提供されている短縮URLサービスを使って <https://git.io/nrpiadminpass.sh> という短縮URLを作りましたので、

```
wget https://git.io/nrpiadminpass.sh
```

でも導入可能です。

スクリプト実行方法

bashスクリプトですので、

```
bash nrpiadminpass.sh パスワード文字列
```

で実行可能です。

パスワード設定スクリプトの取得と実行

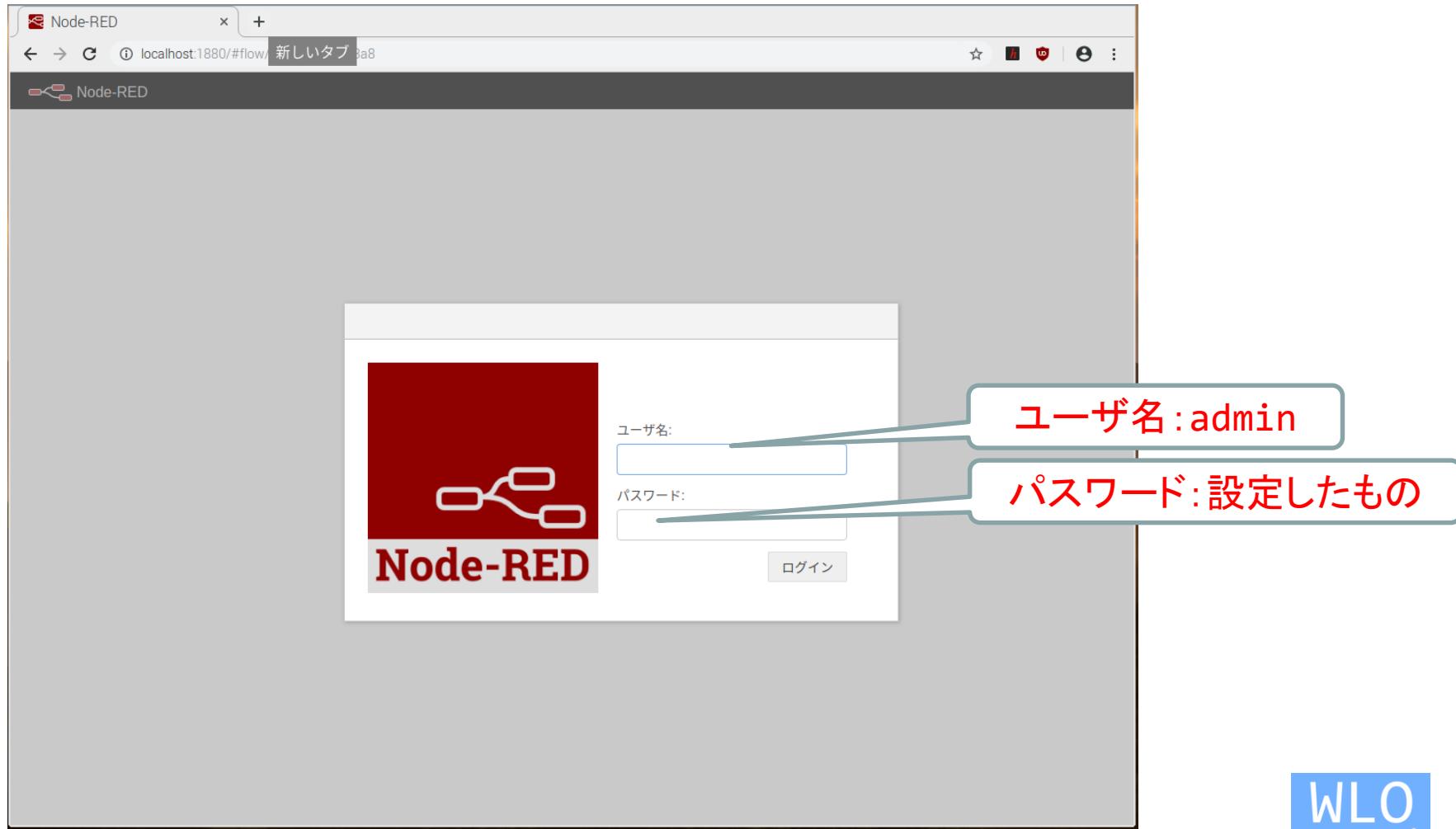
```
$ wget https://git.io/nrpiadminpass.sh ↵
$ bash nrpiadminpass.sh パスワード文字列 ↵
$ node-red-restart ↵
```

パスワードを再設定するときは再度実施

```
$ bash nrpiadminpass.sh パスワード文字列 ↵
$ node-red-restart ↵
```

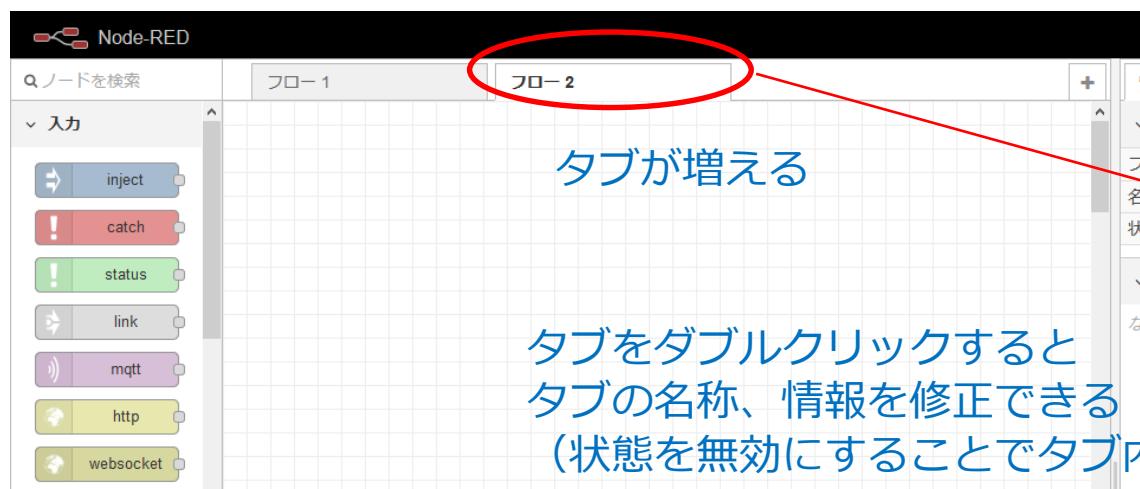
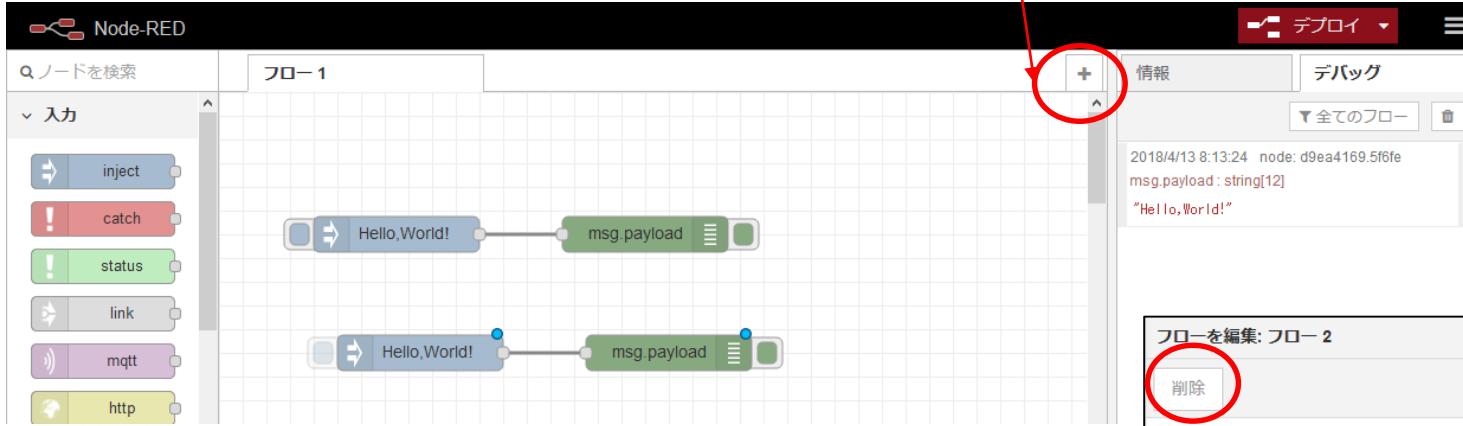
■Node-REDログイン画面の表示とログイン

ブラウザで localhost:1880 にアクセスしてログイン画面が表示されたら成功



■ワークスペースタブの操作

タブの右端「+」をダブルクリック



タブをダブルクリックすると
タブの名称、情報を修正できる
(状態を無効にすることでタブ内のFlowが実行されなくなる)



hands-on 2：ワークスペーススタブ

■コメントの記述

ワークスペースの設定画面で
Markdown形式のコメントが記述できる

フローを編集: フロー 1

削除 中止 完了

名前 フロー 1

状態 有効

詳細

```

1 function nodeのサンプルフロー
2
3 function nodeは以下の処理を行う
4
5 * ``msg.payload``に格納されている文字列の文字数を算出
6 * 算出した文字数を``msg.payload``に上書き保存
    
```

情報 デバッグ

フロー "cfb1a697.495558"
名前 フロー 1
状態 有効

フローの詳細

function nodeのサンプルフロー
function nodeは以下の処理を行う

- msg.payload に格納されている文字列の文字数を算出
- 算出した文字数を msg.payload に上書き保存

記述したコメントは
情報タブに表示される

comment ノードを編集

削除 中止 完了

プロパティ

タイトル payloadの文字数を算出

本文

```

1 以下の処理を行う
2
3 * ``msg.payload``に格納されている文字列の文字数を算出
4 * 算出した文字数を``msg.payload``に上書き保存
    
```

情報 デバッグ

フロー "cfb1a697.495558"
名前 フロー 1
状態 有効

フローの詳細

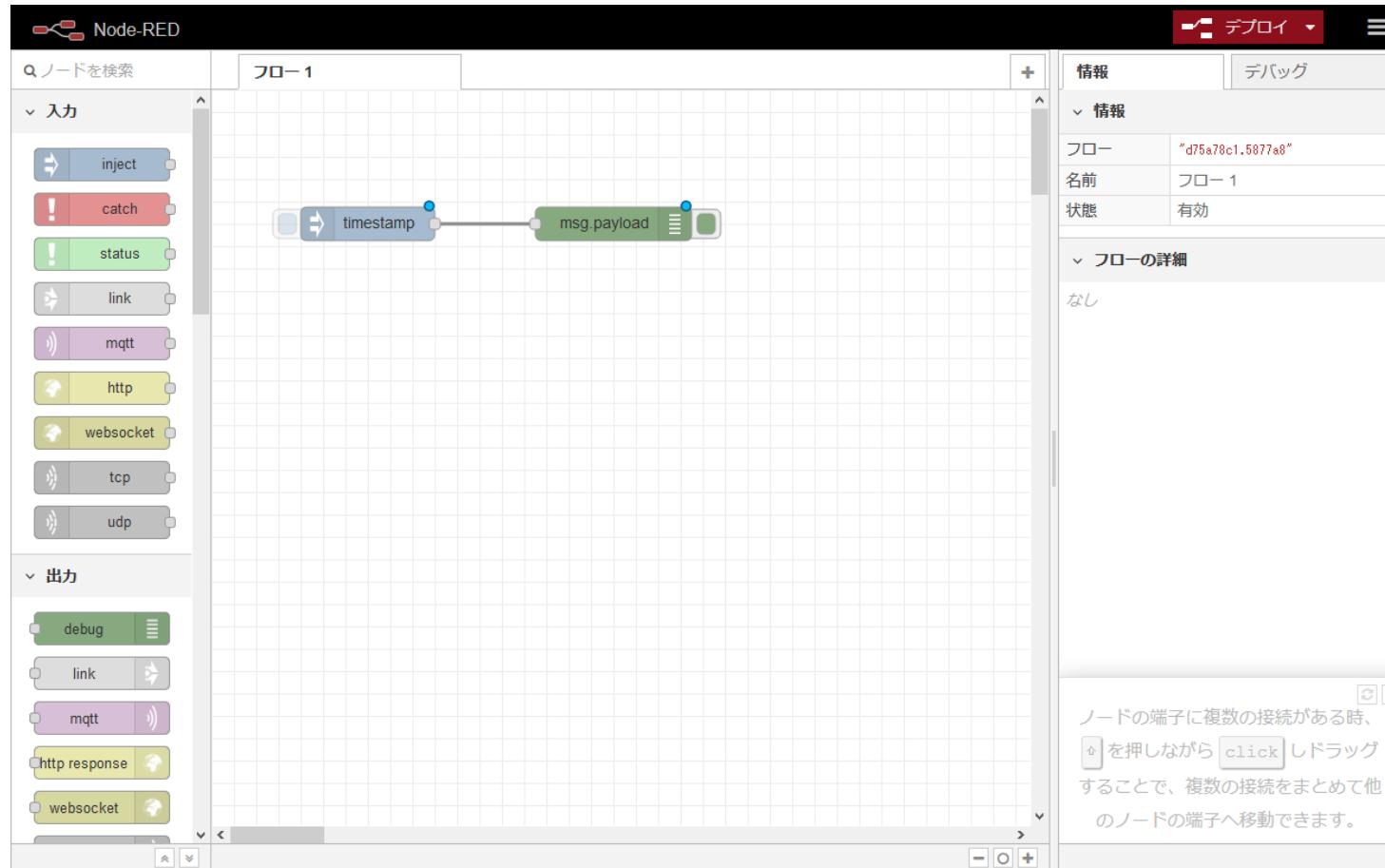
function nodeのサンプルフロー
function nodeは以下の処理を行う

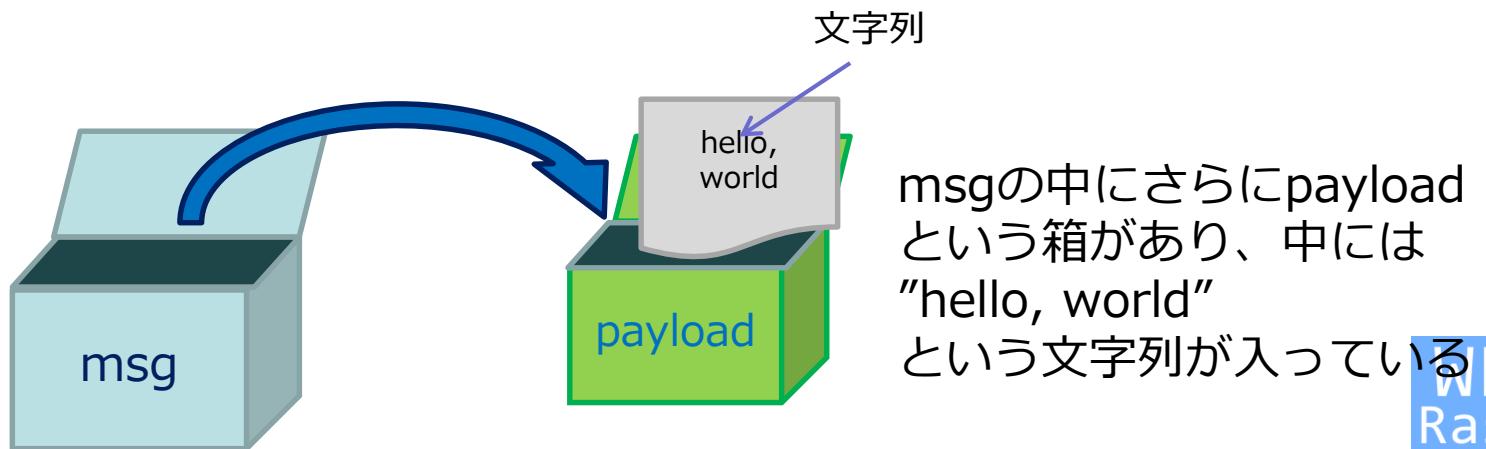
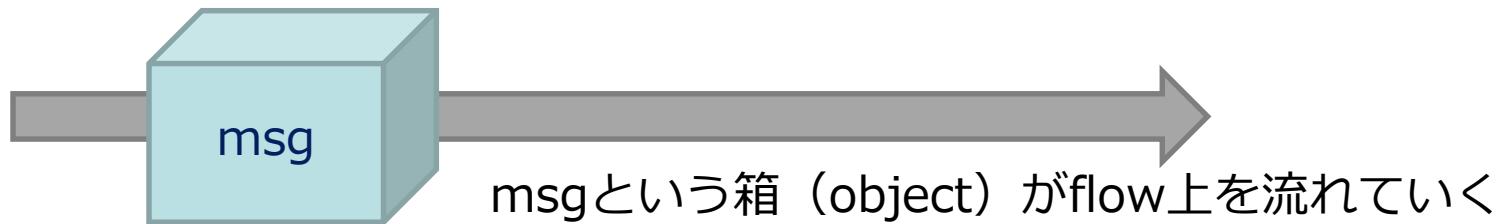
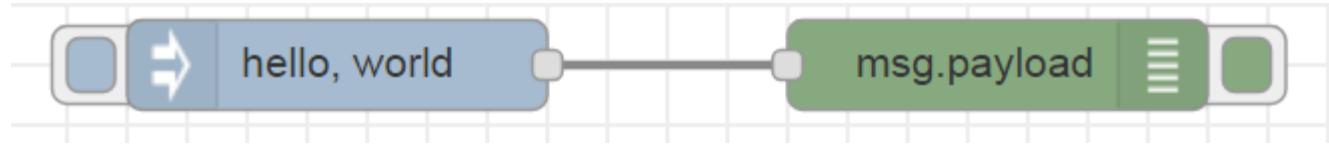
- msg.payload に格納されている文字列の文字数を算出
- 算出した文字数を msg.payload に上書き保存

comment nodeを使えばflow内にコメントを
置くことができる

<http://git.io/wlopi>

■最初のFlow (再掲)





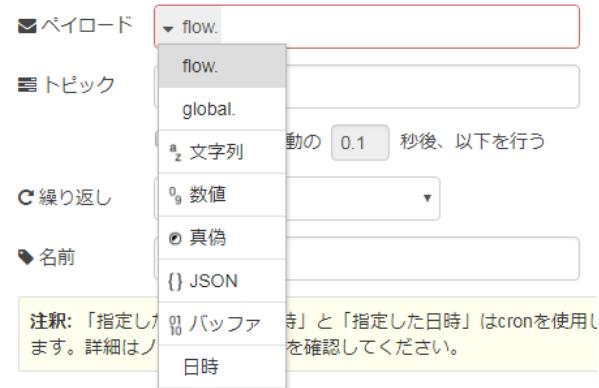
■ inject node



payloadに保存する値を設定
デフォルトではtimestamp（日時のミリ秒）

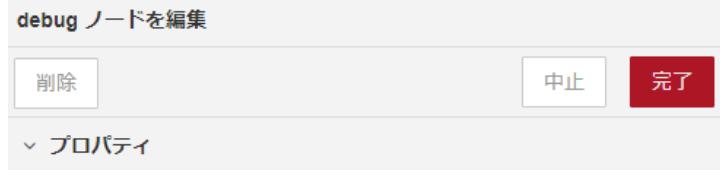
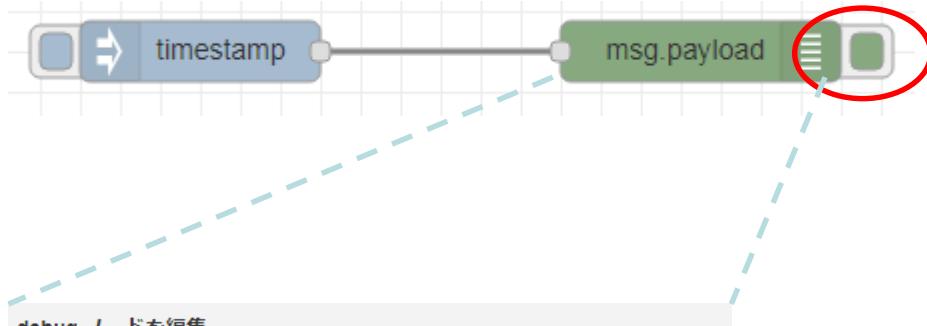
左端をクリックするとmsgがflowを流れる（flowの実行）

nodeをダブルクリックすると
設定画面が表示される

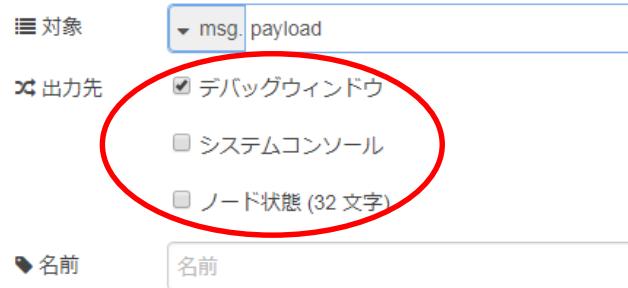


payloadの値は色々選べる
よく使うのは
・文字列
・数値
・JSON

■debug node



右端が縁になっていないと
情報が表示されない
(変更の度にデプロイが必要)



- ・**デバッグウィンドウ :**
チェックを入れるとデバッグタブに表示される
- ・**システムコンソール :**
チェックを入れるとコンソールに表示される
- ・**ノード状態(32文字) :**
チェックを入れるとnodeの下に表示される
(最大32文字)

debug nodeは実行時に
メモリを消費します
使いすぎに注意



■template node

The screenshot shows a Node-RED flow with three nodes: an inject node, a template node, and a msg.payload node. The inject node has a payload of "花子". The template node has a template string: "1 こんにちは、{{payload}}さん". The msg.payload node outputs the result: "こんにちは、花子さん".

inject ノードを編集

削除 中止 完了

▼ プロパティ

■ ペイロード 花子

■ トピック

Node-RED起動の 0.1 秒後、以下を行う

template ノードを編集

削除 中止 完了

▼ プロパティ

■ 名前 例前

■ 設定先 msg. payload

✓ 形式 Mustacheテンプレート

■ テンプレート

1 こんにちは、{{payload}}さん

言語: mustache

A red circle highlights the "花子" payload in the inject node's properties. Another red circle highlights the Mustache template string in the template node's properties.

自分の名前を設定

{{ }}中括弧2つで囲んだ部分がオブジェクトに置き換わる(Mustache記法)

'{{payload}}' で msg.payloadの値を取り出すことができる

The screenshot shows the Node-RED interface with the deployed flow. The logs panel shows the output: "2018/7/24 14:38:57 node: c6c6edc.ccdee1 msg.payload : string[10] "こんにちは、花子さん"".

Node-RED

ノードを検索

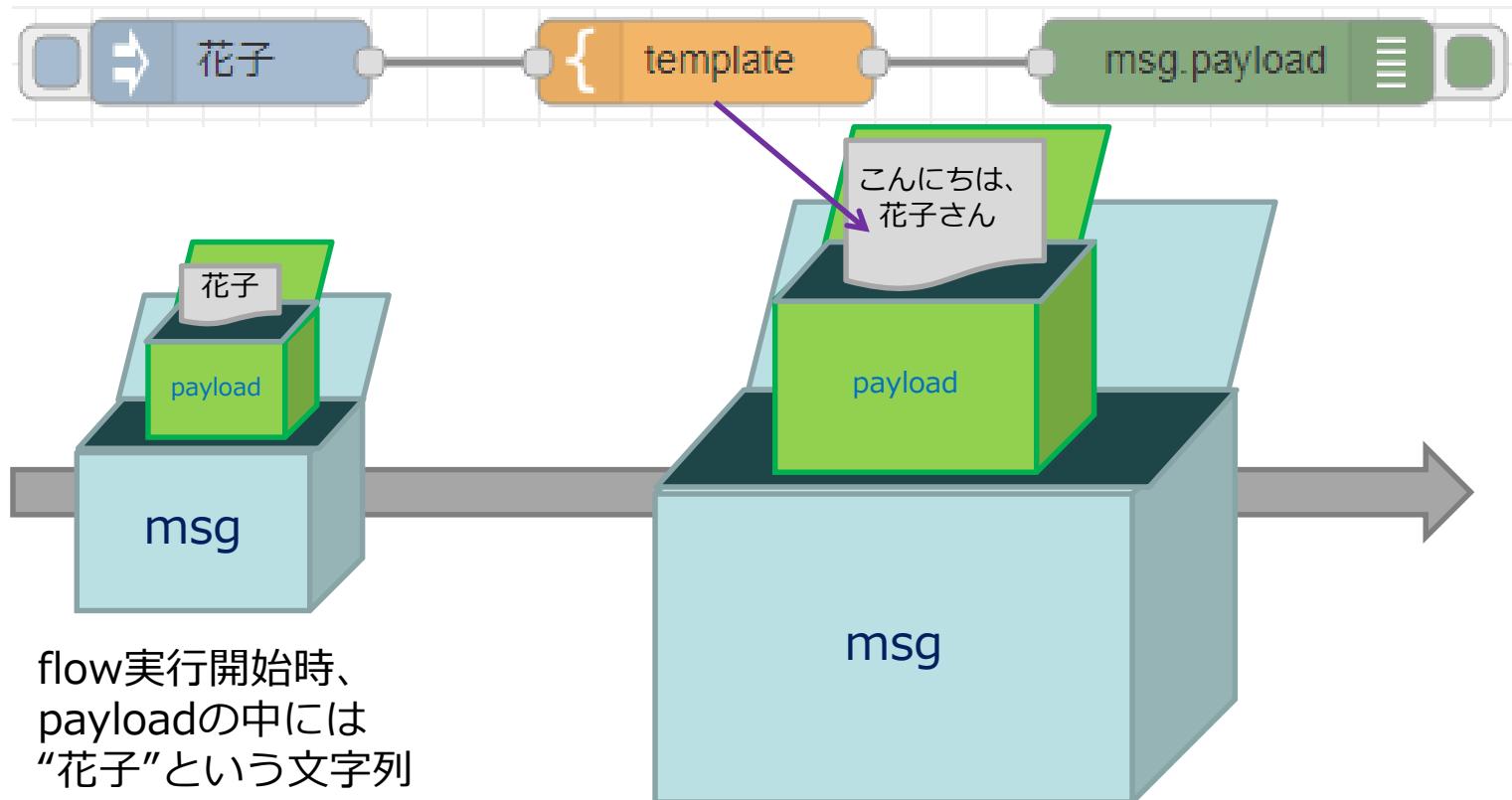
フロー 1

+ 情報 デバッグ

全般的なフロー

2018/7/24 14:38:57 node: c6c6edc.ccdee1
msg.payload : string[10]
"こんにちは、花子さん"

■template node



hands-on 2 : 標準node

■ http in, http response node



`{{payload}}` を
`{{req.params.name}}` に変更

<http://git.io/wlopi>

hands-on 2 : 標準node

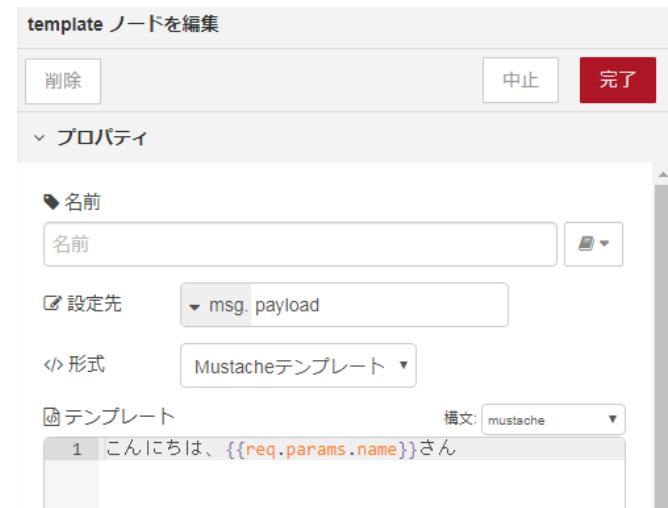
■ http in, http response node

ブラウザで

`http://<Raspberry PiのIPアドレス>:1880/hello/<自分の名前>`
にアクセス

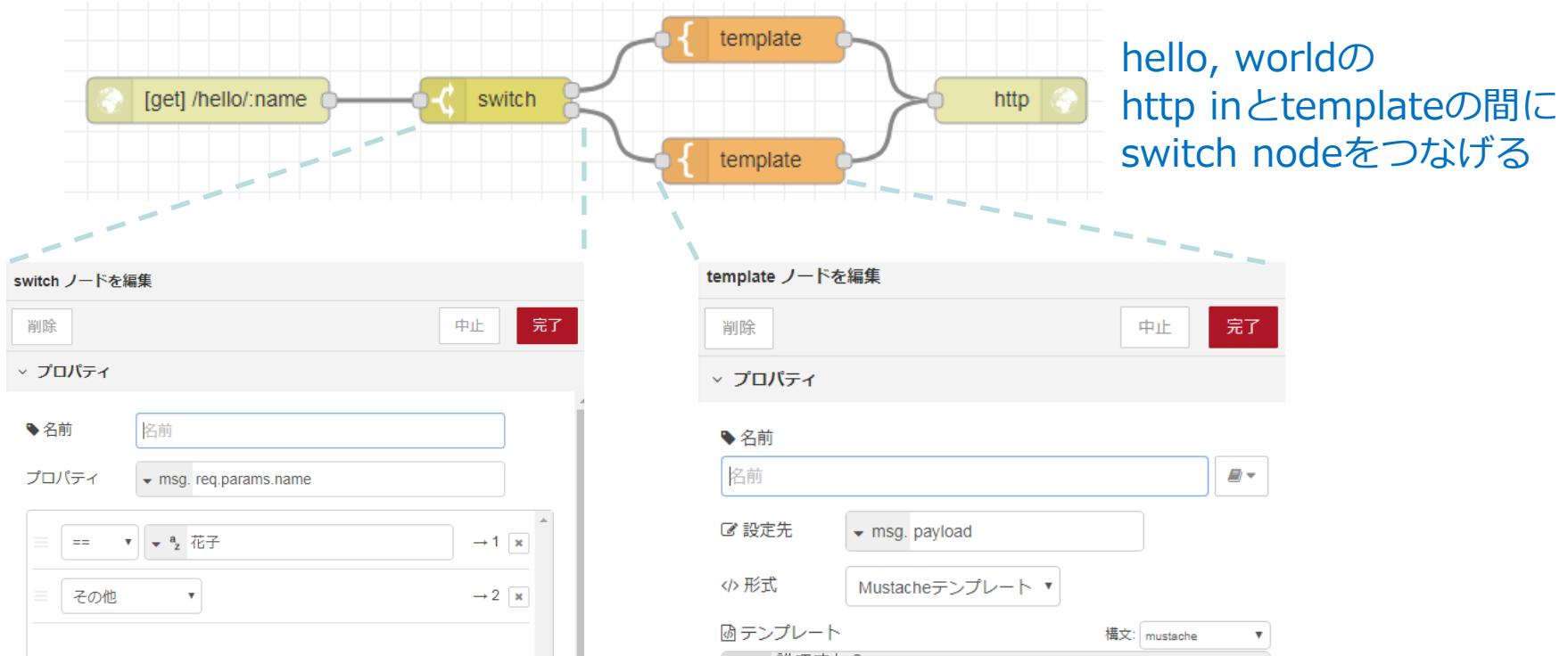


http in nodeでURLに「:」（コロン）付きで設定した部分は
`msg.req.params.<名前>`
で取得できる



`msg.req.params.name`を取り出すには
`{{req.params.name}}`

■switch node



プロパティは msg.req.params.name
分岐は

- ・自分の名前と等しい
- ・その他

を設定

hello, worldの
http inとtemplateの間に
switch nodeをつなげる

switch nodeの下のoutputに
新たなtemplate nodeをつなげる
文字列は「誰ですか？」

hands-on 2： 標準node

■switch node

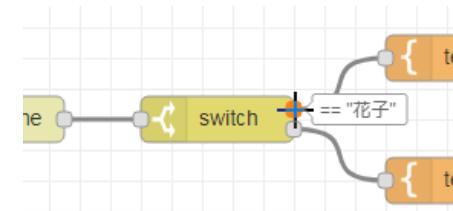


ブラウザで
http://<Raspberry PiのIPアドレス>:1880/hello/<自分の名前>
にアクセス



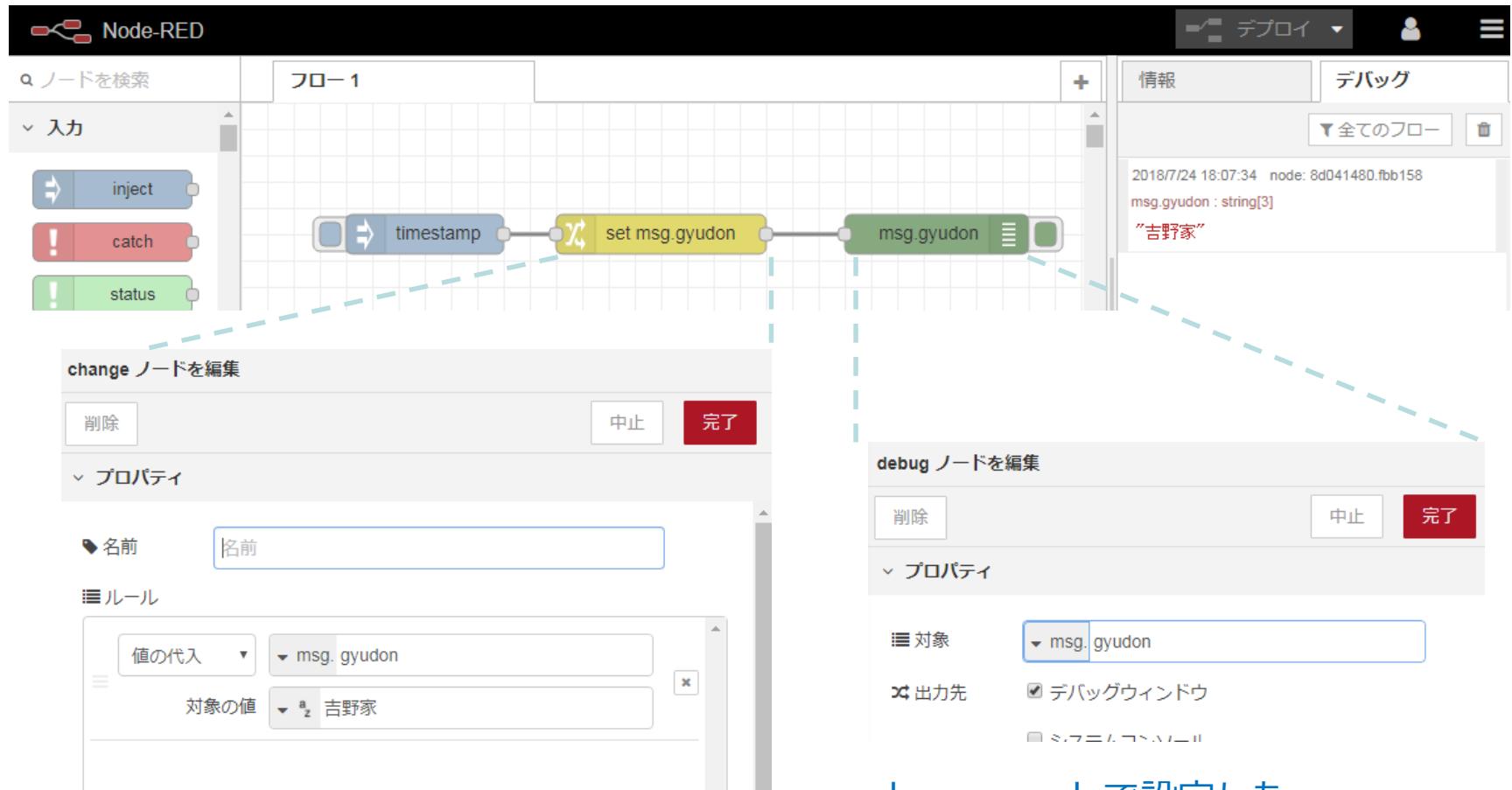
ブラウザで
http://<Raspberry PiのIPアドレス>:1880/hello/<自分の名前以外>
にアクセス

switch nodeの設定で
「最初に合致した条件で終了」
を選択すると無駄な判定処理を
減らすことができる



output（出力）にマウスカーソルを
合わせると、条件が表示される

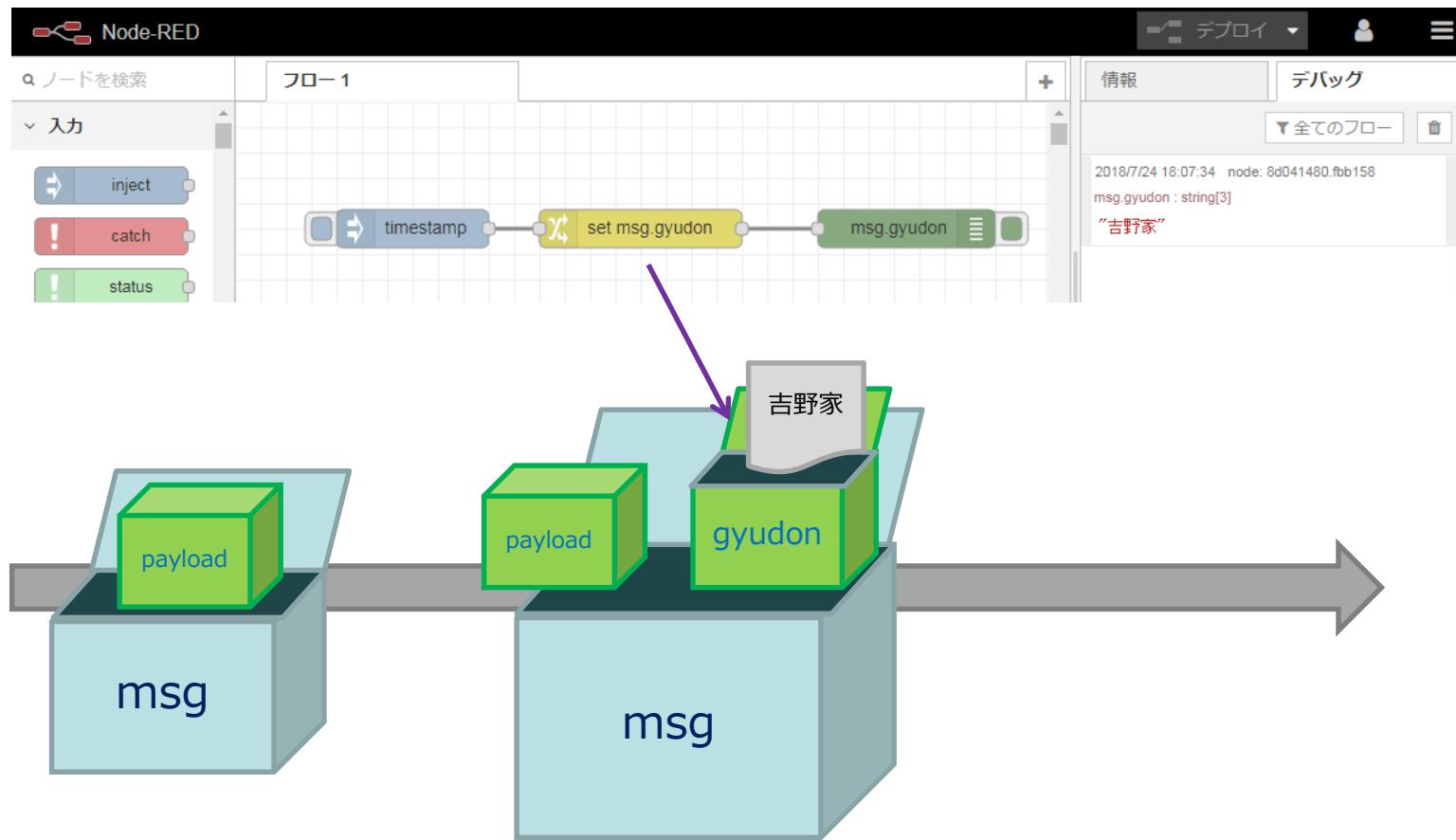
■change node



プロパティに値を代入したり、
あるプロパティから別のプロパティに
コピーや移動ができる

change nodeで設定した
msg.gyudon
の値を表示

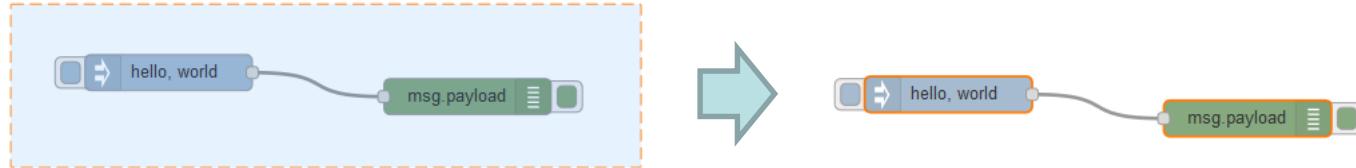
hands-on 2 : flow実行のイメージ



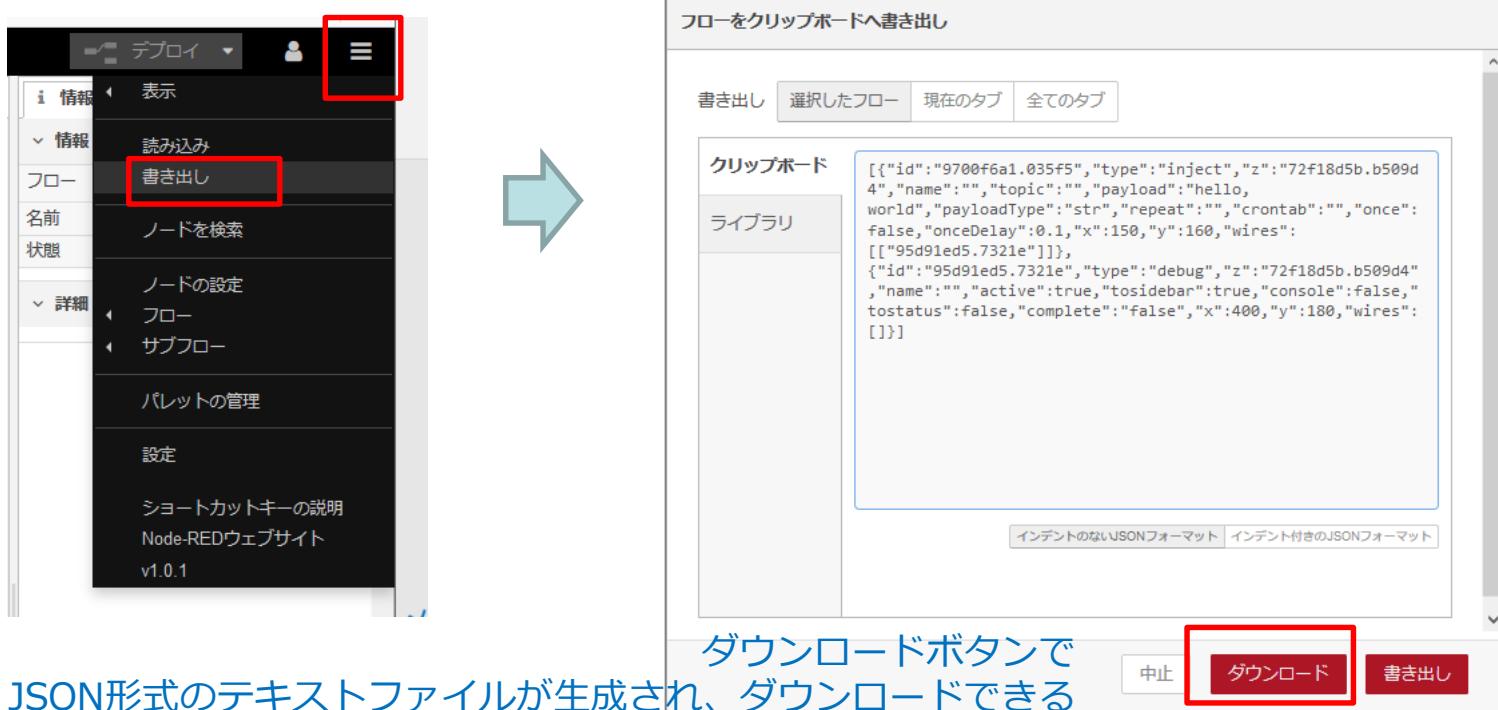
flow実行時はinject nodeで設定したpayloadがmsgの中にいるが
change nodeがさらにgyudonという箱（中身は文字列”吉野家”）を
msgに追加する

■Node-REDのFlowをファイルに書き出す

- 書き出したいFlowをマウスで囲って選択



- FlowEditor右上のハンバーガーメニューの「書き出し」を選択



JSON形式のテキストファイルが生成され、ダウンロードできる



■クリップボード/ファイルからFlowをNode-REDに読み込む

- FlowEditor右上のハンバーガーメニューの「読み込み」を選択

デプロイ 表示

読み込み

書き出し

情報

フロー

名前

状態

詳細

ノードを検索

ノードの設定

フロー

サブフロー

パレットの管理

設定

ショートカットキーの説明

Node-REDウェブサイト

v1.0.1

読み込み先 現在のタブ 新規のタブ

中止 読み込み

読み込みました:
• 2 個のノード

Blockly

読み込み

hello, world

msg.payload

Club

http://git.io/wlopi

ここをクリックして
ファイルで読み込むか

ここにJSON形式の
テキストを
ペースト(Ctrl+V)する



■サンプルFlow : 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

<http://qiita.com/matsuoshi/items/7c19e7dcf404b7d921d6>

The screenshot illustrates the process of finding a Node-RED flow from a GitHub repository:

- Top Left:** A screenshot of the GitHub organization page for "WLO RaspiClub". It shows 25 repositories, 5 packages, and 5 people.
- Middle Left:** A screenshot of the GitHub repository page for "20191010_Node-RED". It describes an event on October 10, 2019, and lists 0 stars, 0 forks, and 0 issues.
- Middle Right:** A screenshot of the GitHub repository page for "utaa ni" showing the "祝日.json" file. The file description is "日本の祝日を取得するNode-REDのFlow".
- Bottom Right:** A screenshot of the GitHub code view for the "祝日.json" file. It shows the JSON content:

```
[{"id": "69bed636.f49078", "type": "inject", "z": "d26a093e.a6a3a8", "name": "", "topic": "", "payload": "", "payloadType": "date", "repeat": ""}]
```
- Bottom Left:** A zoomed-in view of the JSON code, showing the first line:

```
1 [{"id": "69bed636.f49078", "type": "inject", "z": "d26a093e.a6a3a8", "name": "", "topic": "", "payload": "", "payloadType": "date", "repeat": ""}]
```

全選択（左トリプルクリック）→コピー（Ctrl + C）



■サンプルFlow : 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

<http://qiita.com/matsuoshi/items/7c19e7dcf404b7d921d6>

```
1 lines (1 sloc) | 921 Bytes
Raw Blame History
1 [{"id": "69bed636.f49078", "type": "inject", "z": "d26a093e.a6a3a8", "name": "", "topic": "", "payload": "", "payloadType": "date", "repeat": "", "x": 100, "y": 160, "wires": [[{"id": "85e5bf00.8449c"}]]}, {"id": "85e5bf00.8449c", "type": "http request", "z": "d26a093e.a6a3a8", "name": "", "method": "GET", "ret": "obj", "paytoqs": false, "url": "https://holidays-jp.github.io/api/v1/date.json", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 400, "y": 160, "wires": [[{"id": "d8b5be58.a98db", "a924cda.8c9b53"}]]}, {"id": "a924cda.8c9b53", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tostidebar": true, "console": false, "tostatus": false, "complete": "payload.2019-10-22", "targetType": "msg", "x": 550, "y": 220, "wires": []}, {"id": "d8b5be58.a98db", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tostidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "x": 550, "y": 160, "wires": []}]
```

全選択 (左トリプルクリック) → コピー (Ctrl + C)

フローをクリップボードから読み込み

クリップボード JSON形式のフローデータを貼り付けてください
読み込むファイルを選択してください

```
{"id": "69bed636.f49078", "type": "inject", "z": "d26a093e.a6a3a8", "name": "", "topic": "", "payload": "", "payloadType": "date", "repeat": "", "x": 100, "y": 160, "wires": [[{"id": "85e5bf00.8449c"}]]}, {"id": "85e5bf00.8449c", "type": "http request", "z": "d26a093e.a6a3a8", "name": "", "method": "GET", "ret": "obj", "paytoqs": false, "url": "https://holidays-jp.github.io/api/v1/date.json", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 400, "y": 160, "wires": [[{"id": "d8b5be58.a98db", "a924cda.8c9b53"}]]}, {"id": "a924cda.8c9b53", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tostidebar": true, "console": false, "tostatus": false, "complete": "payload.2019-10-22", "targetType": "msg", "x": 550, "y": 220, "wires": []}, {"id": "d8b5be58.a98db", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tostidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "x": 550, "y": 160, "wires": []}]
```

読み込み先 現在のタブ 新規のタブ

中止 読み込み

ペースト (Ctrl + V)

好きな位置にFlowを配置する



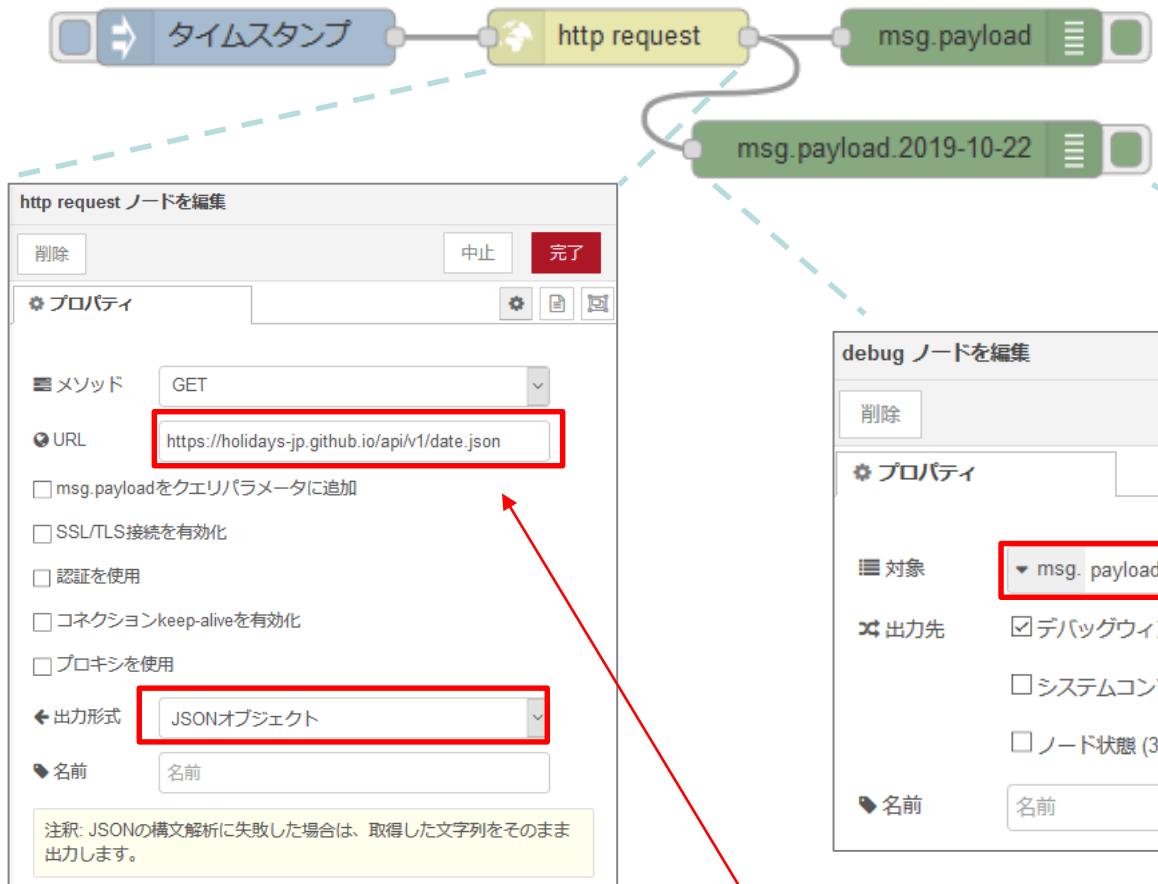
<http://git.io/wlopi>



■サンプルFlow : 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

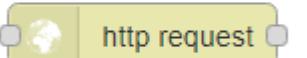
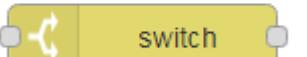
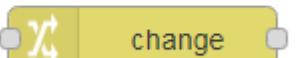
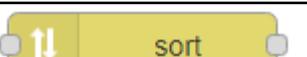
<http://qiita.com/matsuoshi/items/7c19e7dcf404b7d921d6>

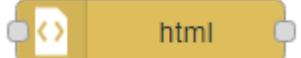
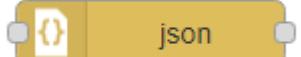
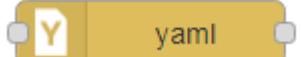


<https://holidays-jp.github.io/api/v1/date.json>

<http://git.io/wlopi>

▼ 入力	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。
 link	標準	link outのNodeとセットで用いることで、Flowをリンクでつなげる事ができる。Tabを跨いで使用することができる。
 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	指定URL (ws://XXX) で受信したパケットを、メッセージのデータとしてFlowに出力する。websocket outのNodeとセットで用いる。
▼ 出力		
 debug	標準	メッセージのデータをDebugタブやシステムコンソールに出力できる。ステータステキストに短い文字（32文字）を出力することもできる。
 link	標準	link inのNodeとセットで用いることで、Node間をリンクでつなげる事ができる。Tabを跨いで使用することができる。
 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 in のNodeとセットで用いる。
 websocket	標準	メッセージのデータを指定URL (ws://XXX) のクライアントに送信する。websocket inのNodeとセットで用いる。

機能	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。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。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。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する。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。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。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。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。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。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。
 split	標準 メッセージを分割する。Function Nodeでも記載可能だがより視覚的。
 join	標準 メッセージを連結する。Function Nodeでも記載可能だがより視覚的。
 sort	標準 配列の並び替えを行う。昇順/降順を指定できる。

機能	標準orカスタム
 batch	標準 メッセージをグループ化して、まとめて出力するNode。グループ化するメッセージの数を指定できる。
 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。
 yaml	標準 yaml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。
 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也可能。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。

ストレージ	標準orカスタム
 tail	標準
 file	標準
 file	標準
分析	
 sentiment	標準
その他	
 watch	標準
 feedparse	標準
 exec	標準



■カスタムNodeは以下のサイトで検索できます

<https://flows.nodered.org/search?type=node>

The screenshot shows a search results page for 'nodes' on the flows.nodered.org website. The results are displayed in a grid of cards:

- node-red-contrib-sensibo: Communication nodes to Sensibo cloud (v0.2.0)
- red-node-incorta: A Node-RED node to read and write to a galan database using JDBC (v0.1.1)
- node-red-contrib-alexa-remote2: node-red nodes for interacting with alexa (v3.2.8)
- node-red-contrib-msg-router: A Node Red node to route messages between nodes (v0.0.6)
- node-red-contrib-tibbo-ltps-tibbits: LTPS Tibbit nodes for node-red (v0.0.6)
- node-red-node-ui-list: Node-RED Dashboard UI widget node for simple list (v0.2.4)
- node-red-contrib-winccoa: WinCC OA to Node-RED connector (v2.0.5)
- node-red-contrib-dockerode: node-red nodes to communicate with docker (v0.3.3)
- node-red-contrib-google-home-notifier-offline: Node-Red nodes for google-home-notifier-offline library. With this node you can cast any (v0.1.0)
- node-red-contrib-grove-base-hat: The node for Grove Base HAT for Raspberry Pi. (v0.0.5)
- node-red-contrib-nordpool-api: A Node-RED Node for collecting "day ahead" prices from NordpoolGroup. (v2.0.0)
- node-red-contrib-meraki-dashboard-api: Node-RED node for meraki-dashboard-api (v0.4.2)
- ncd-red-wireless: (!)Build Status (https://semaphoreci.com/api/v1) (v0.0.0)
- @ia-cloud/node-red-contrib-ia-cloud-fds-raspberry-pi: (v0.0.0)
- enact-actuation-conflict-manager-node: (v0.0.0)

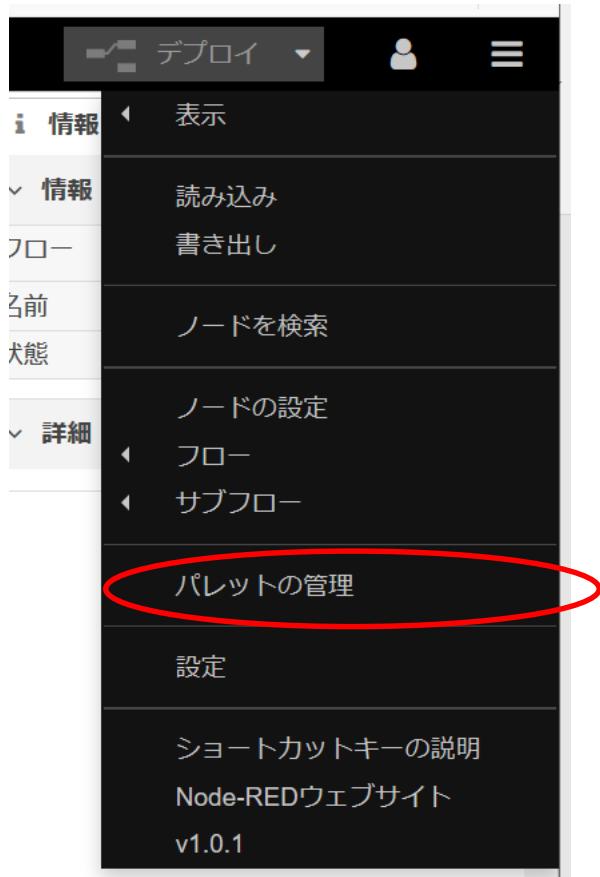
The screenshot shows the Node-RED interface with a sidebar menu. The 'パレットの管理' (Palette Management) option is highlighted with a red circle.

- デプロイ
- 情報
- 読み込み
- 書き出し
- ノードを検索
- ノードの設定
- フロー
- サブフロー
- パレットの管理
- 設定
- ショートカットキーの説明
- Node-REDウェブサイト
- v0.18.7

メニューの
「パレットの管理」



■カスタムノードの追加



ハンバーガーメニューの
「パレットの管理」

ノード名	バージョン	ノード数	操作
node-red	0.18.4	> 50 個のノード	削除 全て無効化
node-red-contrib-aws-iot-hub	0.1.12	> 4 個のノード	削除 全て無効化
node-red-contrib-aws-sdk	0.1.7	> 2 個のノード	削除 全て無効化
node-red-contrib-sonospollytts	1.0.1	> 2 個のノード	削除 全て無効化
node-red-node-email	0.1.27	> 2 個のノード	削除 全て無効化
node-red-node-feedparser	0.1.12	> 1 個のノード	削除 全て無効化
node-red-node-rbe	0.2.3	> 1 個のノード	削除 全て無効化
node-red-node-twitter			

「現在のノード」タブで
インストールされている
ノードの一覧を確認
(削除、更新も可能)

<http://git.io/wlopi>



■ Lineに通知を送る「Line Notify」のカスタムノードを導入してみる。

「ノードを追加」タブで「node-red-contrib-line-notify」で検索



Node-RED

Search library

node-red-contrib-line-notify 1.0.3

line notify node

```
npm install node-red-contrib-line-notify
```

An easy-to-use NodeRED node for sending Line Notify.

簡単にライン通知を送ることができます。

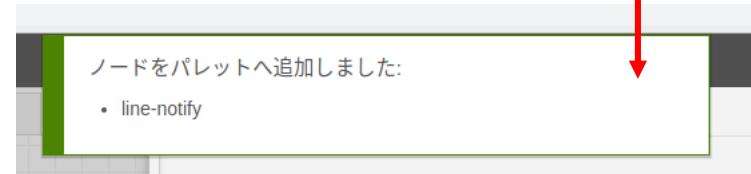
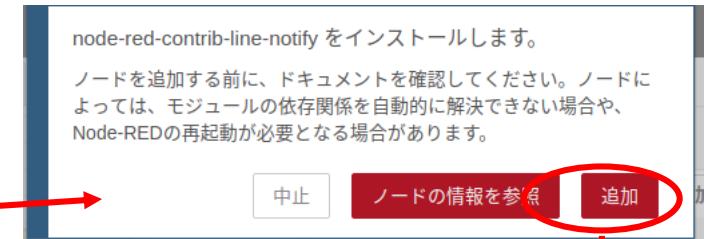
Install

Run the following command in your Node-RED user directory - typically `~/.node-red`

下記のコマンドでインストールください。

```
npm install node-red-contrib-line-notify
```

ノードの説明ページを参照



カスタムノードが
パレットに
追加される
<http://git.io/wlopi>



■ Line Notify

LINE Notifyにログイン

<https://notify-bot.line.me/ja/>



LINEアプリで友達に追加

トークンを発行する

トークン名を記入してください (通知の際に表示されます)

Playground

通知を送信するトグルームを選択してください

Search by group name

1:1でLINE Notifyから通知を受け取る

トークン名: Playground

「1:1でLINE Notifyから通知を受け取る」を選択し
「発行する」をクリック

発行したトークンはこちらです。

PFNefcHvJgjAeL... (copy) (cancel)

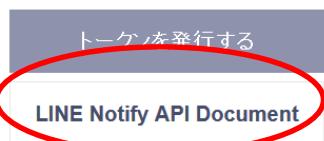
このページから移動すると、新しく発行されたトークンは二度と表示されません。トークンをコピーしてからページを移動して下さい。

コピー

閉じる

アクセストークンの発行(開発者向け)

パーソナルアクセストークンを利用することで、Webサービスの登録をせずに通知を設定することができます。



トークンを発行する

LINE Notify API Document

右上のメニューから
「マイページ」を選択
「トークンを発行する」をクリック

トークンはここでしか
表示されないので
コピーしておく

<http://git.io/wlopi>



■node-red-contrib-line-notify

change ノードを編集

削除 中止 完了

※ プロパティ

名前: 名前

ルール

- 値の代入: msg. contentType → 対象の値: message
- 値の代入: msg. message → 対象の値: Node-REDからこんにちは

+ 追加

msg.contentType→「message」
msg.message→送信する文字列

左下の「追加」を押して
変更項目を追加する

line-notify ノードを編集

削除 中止 完了

※ プロパティ

Name: Name

アクセストークン: (Redacted)

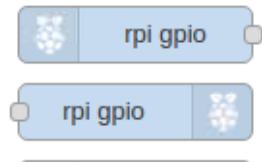
コンテンツタイプ: メッセージ

前頁で取得したトークンを
ここにペーストする



■スイッチ用GPIOinノードの設定

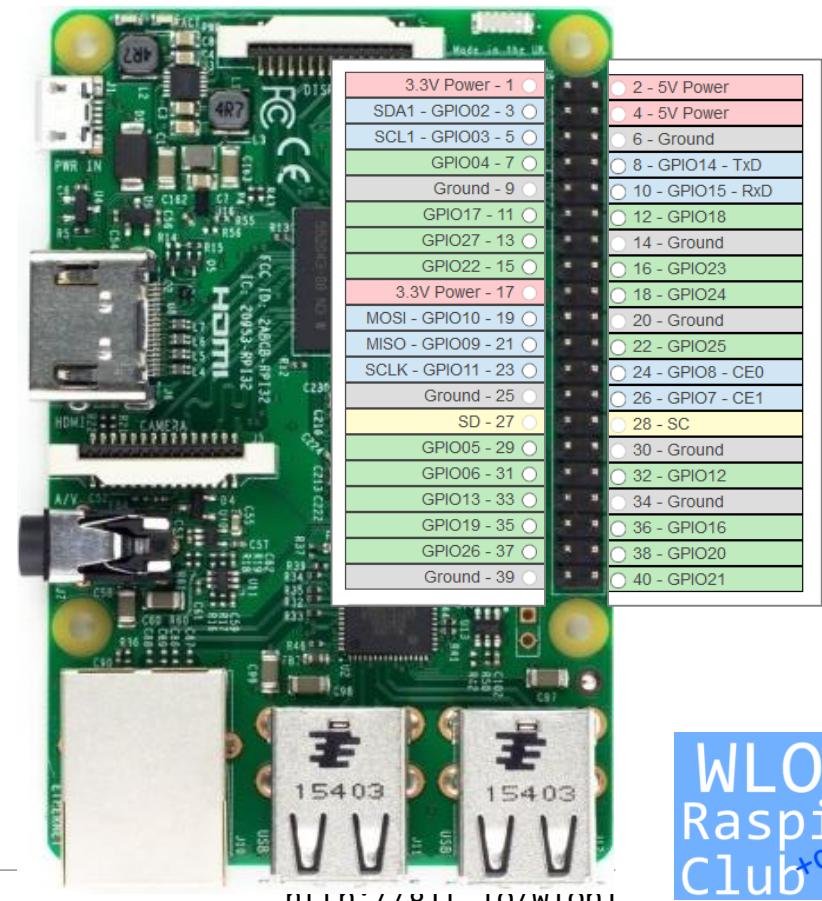
▼ Raspberry Pi



GPIO in / GPIO out ノード

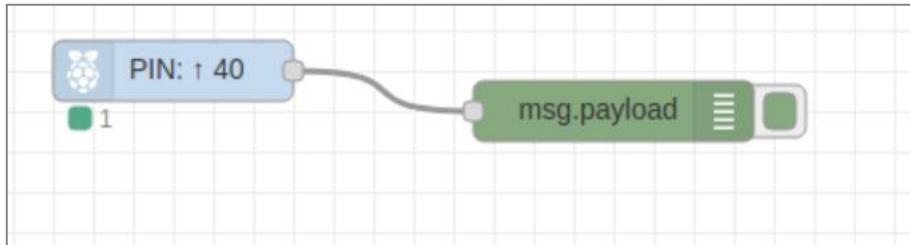
※Raspberry PiのNode-REDには標準導入されています

GPIO in: スイッチ入力

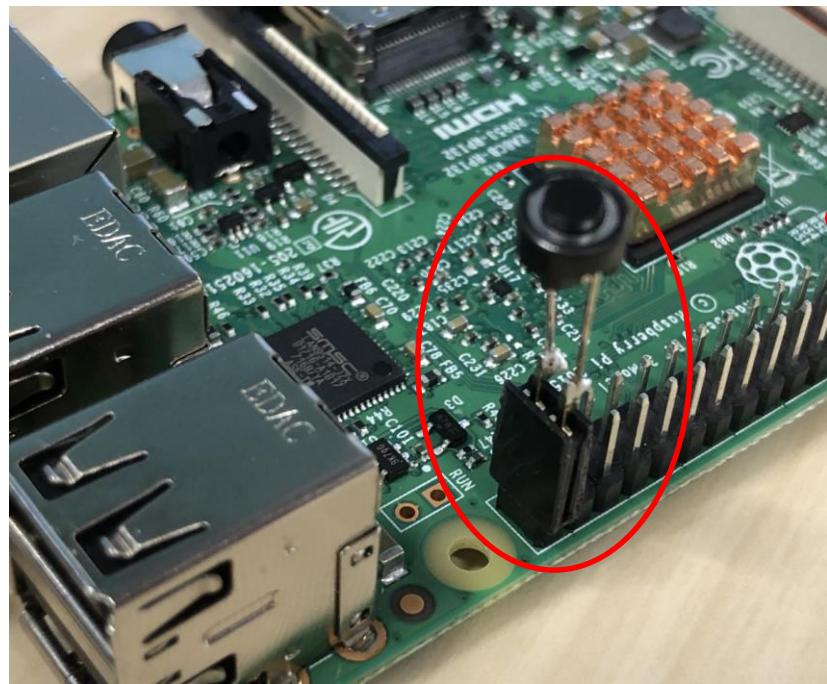




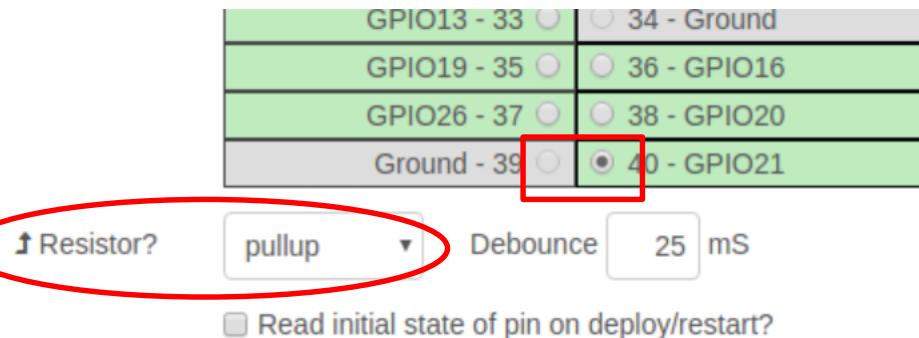
■スイッチ用GPIOinノードのFlow (基本)



スイッチが押されたら0が出力
スイッチが離れたら1が出力



GPIOの39pinと40pin (一番Ethernet側)に
プッシュスイッチを刺す



スイッチが押されたら、
GroundとGPIO21がつながる
→0が出力

スイッチがはなれたら、
GroundとGPIO21が離れる
→ResistorでPullupとしておくと
3.3Vになる
→1が出力



■スイッチ用GPIOinノードのFlow (応用)

スイッチが押されたときだけ1が出力

rangeノード：
値の範囲を入れ替える

rbeノード：
例外データを検知する

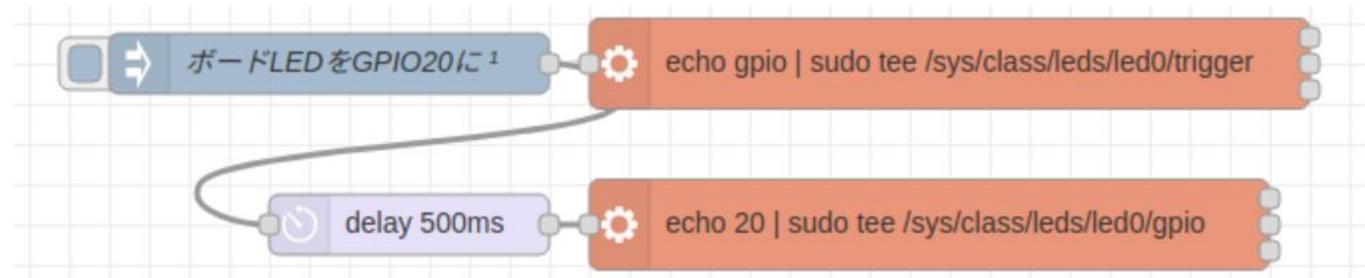
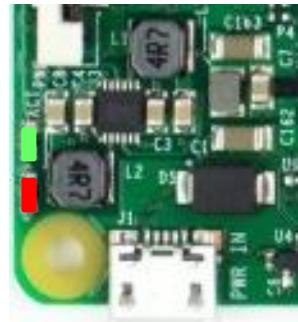


■ボードのステータスLED(緑) をGPIOoutノードで操作

Raspbianの設定を行うことで、
ステータスLED（緑）をGPIOで操作できるようになる

ステータスLED→

参考：メモ：Raspberry Piの基板上のLEDを別の用途に使う
<https://qiita.com/spicemanjp/items/fbe812de4f2b6b3ebbb6>



■ボードのステータスLED(緑) をGPIOoutノードで操作



A screenshot of a GitHub code editor showing the 'raspberrypiboardled.json' file. The file content is as follows:

```
[{"id": "5fc1cf3a.d659e", "type": "exec", "z": "d62ca71d.243728", "command": "echo gpio | sudo tee /sys/class/leds/led0/trigger", "addpay": false, "pay": ""}]
```

The code editor has a toolbar with 'Raw', 'Blame', and 'History' buttons. Below the code, there is a preview pane showing the JSON structure.

全選択 (左トリプルクリック) → コピー (Ctrl + C)



■ ボードのステータスLED(緑) をGPIOoutノードで操作

```
1 lines (1 sloc) | 921 Bytes
Raw Blame History
1 [{"id": "69bed636.f49078", "type": "inject", "z": "d26a093e.a6a3a8", "name": "", "topic": "", "payload": "", "payloadType": "date", "repeat": "", "x": 100, "y": 160, "wires": [[{"id": "85e5bf00.8449c"}]]}, {"id": "85e5bf00.8449c", "type": "http request", "z": "d26a093e.a6a3a8", "name": "", "method": "GET", "request": "", "obj": "payload", "paytoqs": false, "url": "https://holidays-jp.github.io/api/v1/date.json", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 400, "y": 160, "wires": [[{"id": "d8b5be58.a98db", "a924cda.8c9b53"}]]}, {"id": "a924cda.8c9b53", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload.2019-10-22", "targetType": "msg", "x": 550, "y": 220, "wires": []}, {"id": "d8b5be58.a98db", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "x": 590, "y": 160, "wires": []}], [{"id": "69bed636.f49078", "type": "inject", "z": "d26a093e.a6a3a8", "name": "", "topic": "", "payload": "", "payloadType": "date", "repeat": "", "x": 100, "y": 160, "wires": [[{"id": "85e5bf00.8449c"}]]}, {"id": "85e5bf00.8449c", "type": "http request", "z": "d26a093e.a6a3a8", "name": "", "method": "GET", "request": "", "obj": "payload", "paytoqs": false, "url": "https://holidays-jp.github.io/api/v1/date.json", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 400, "y": 160, "wires": [[{"id": "d8b5be58.a98db", "a924cda.8c9b53"}]]}, {"id": "a924cda.8c9b53", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload.2019-10-22", "targetType": "msg", "x": 550, "y": 220, "wires": []}, {"id": "d8b5be58.a98db", "type": "debug", "z": "d26a093e.a6a3a8", "name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "x": 590, "y": 160, "wires": []}]]
```

全選択 (左トリプルクリック) → コピー (Ctrl + C)

フローをクリップポートから読み込み

クリップボード

ライブラリ

サンプル

読み込むファイルを選択してください

JSON形式のフローデータを貼り付けてください

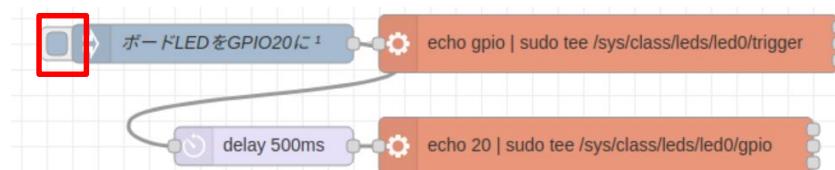
読み込み先 現在のタブ 新規のタブ 中止 読み込み

ペースト (Ctrl + V)

好きな位置にFlowを配置する



インジェクトすると実行完了





■ ボードのステータスLED(緑) をGPIOoutノードで操作

38Pin(GPIO20)を選択して
Digital outputを選択

inject ノードを編集

削除 中止 完了

＊ プロパティ

＊ ペイロード flow. global.

＊ トピック

＊ 文字列 動的 秒後、以下を行う

＊ 数値

＊ 真偽

＊ JSON

注釈: 「指定した日時」と「指定した日時」はcronを使用します。
「時間間隔」に小さな値を指定します。
詳しくはノードの説明文を参照してください。

数値を指定して
0と1を入れる

rpi-gpio out ノードを編集

削除 中止 完了

＊ プロパティ

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)