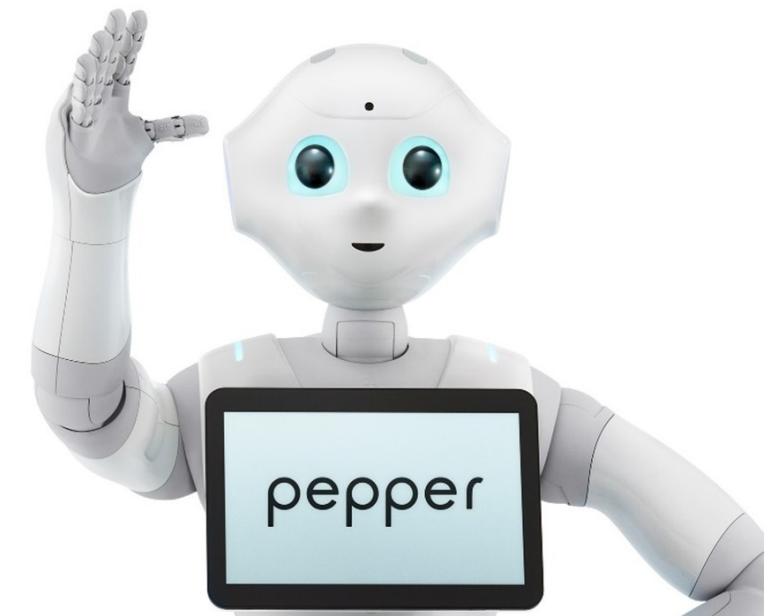




IBM Bluemix を Pepper とつないでみよう

Pepper x Bluemixハンズオンワークショップ&デモ

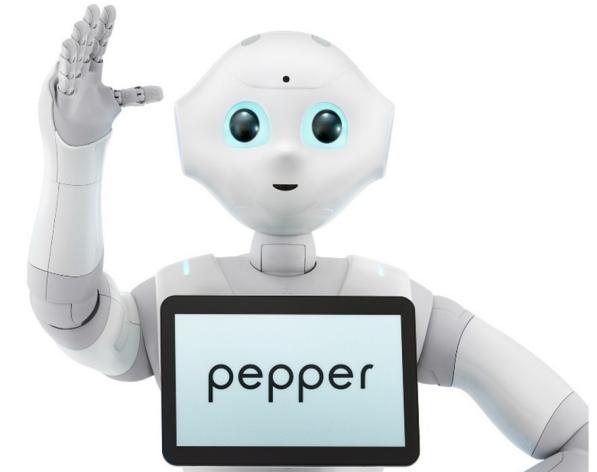
2016/10/31
株式会社スマートロボティクス



内容

- IBM Bluemixとの連携その様々な手法
- Node-RED を使って Pepper と Bluemix を連携
 - HTTPリクエスト(1)単純な HTTP リクエスト
 - HTTPリクエスト(2)画像データの送信
(Watson Visual Recognition で画像認識)
 - HTTPリクエスト(3)音声データの送信
(Watson Speech To Text で音声認識)
 - HTTPリクエスト(4)データの保存
 - WebSocket 通信
 - 外部サービスを Node-RED 経由で利用
- Bluemix 各サービスAPIに直接アクセス
 - 例：Watson Speech To Text の Websocket インターフェースを使う
- サービス提供の SDK を使う
 - Watson SDK
- Pepper のタブレット

IBM Bluemixとの連携 その様々な手法



Pepper とのサービス連携

- Web API にアクセス or サービス提供のSDK を使う？
 - Web API にアクセス
 - REST API が提供されている場合、HTTP リクエストを用いてクラウドサービスと連携する
 - その他、連携のためのプロトコルが公開されている場合、それらを用いてクラウドサービス連携する
 - WebSocket
 - MQTT
 - など
 - サービス提供の SDK をプロジェクトを使う
 - Linux で稼働する Python SDK があれば Pepper で利用できる可能性高い
 - それぞれの SDK に仕様上の特性あり。使いこなせれば開発が楽

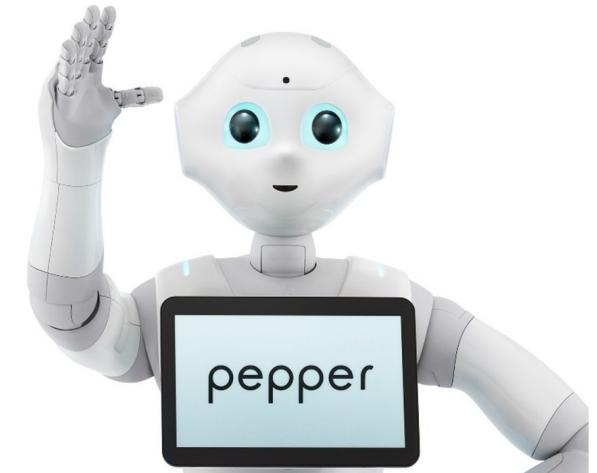
IBM Bluemix と Pepper の連携

手法	連携方法	サービスを利用するための認証	サービスを利用するロジック	適用場面
各サービスの API に直接アクセス	Web API にアクセス	Pepper が行う	Pepper 側で実装	ミニマムのシステム構成でシステムを構築したい場合。API の機能を最大限利用したい場合
Node-RED を使う		Node-RED が行う	Node-RED 側と Pepper 側で分散可能	ラピッドプロトタイピングに適したソリューション
サービス提供の SDK を使う	サービス提供の SDK をプロジェクトに取り込む	Pepper が行う (SDK を使って)	Pepper 側で実装	<ul style="list-style-type: none">Watson Developer Cloud Python SDKBluemix 全体に関する Python での SDK 提供はない。

IBM Bluemix と Pepper の連携

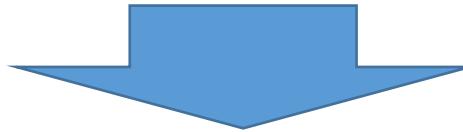
手法	連携方法	サービスを利用するための認証	サービスを利用するロジック	適用場面
各サービスの API に直接アクセス	Web API にアクセス	Pepper が行う	Pepper 側で実装	ミニマムのシステム構成でシステムを構築したい場合。API の機能を最大限利用したい場合
Node-RED を使う		Node-RED が行う	Node-RED 側と Pepper 側で分散可能	ラピッドプロトタイピングに適したソリューション
サービス提供の SDK を使う	サービス提供の SDK をプロジェクトに取り込む	Pepper が行つ (SDK を使って)	Pepper 側で実装	<ul style="list-style-type: none">watson Developer Cloud Python SDKBluemix 全体に関する Python での SDK 提供はない。

Node-RED を使って Pepper と Bluemix を連携



Pepper から Web API へのアクセス

- Python 標準ライブラリ or 外部モジュール利用
- 標準ライブラリを使った場合実装が煩雑になりがち



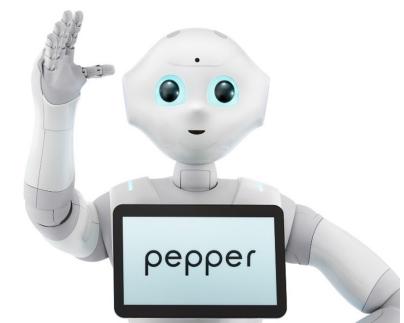
- 今回は HTTP アクセスのモジュールとして requests、WebSocket のモジュールとして liris/websocket-client を利用

サンプルアプリケーション

プロジェクト名	概要
01simplehttprequest	単純な HTTP リクエスト
02uploadimage	画像を Node-RED に送る
03uploadaudio	音声データを Node-RED に送る
04downloadfile	バイナリデータを Node-RED からダウンロードする
05websocket	WebSocket で通信をする
06useotherservice	Node-RED を介して外部 WebAPI を使う

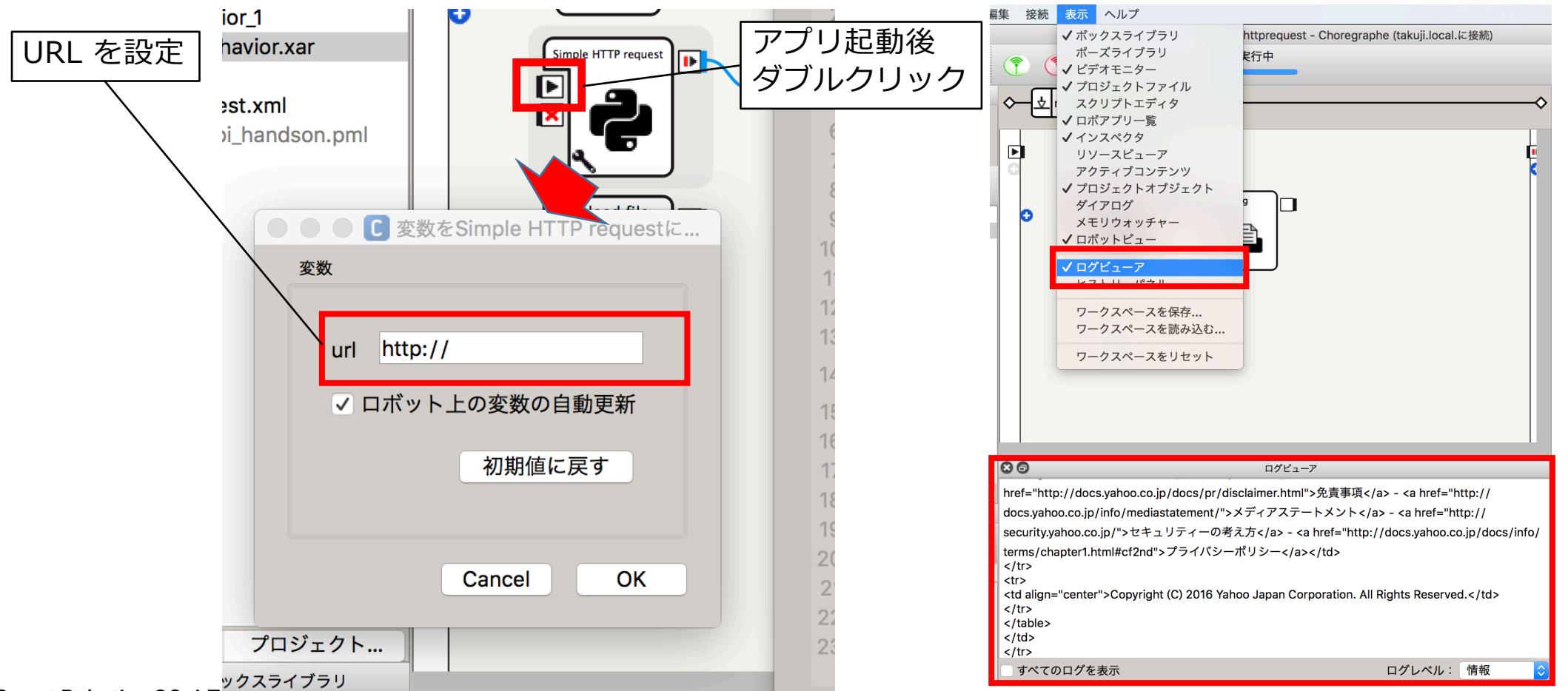
HTTPリクエスト(1) 単純な HTTP リクエスト

01simplehttprequest



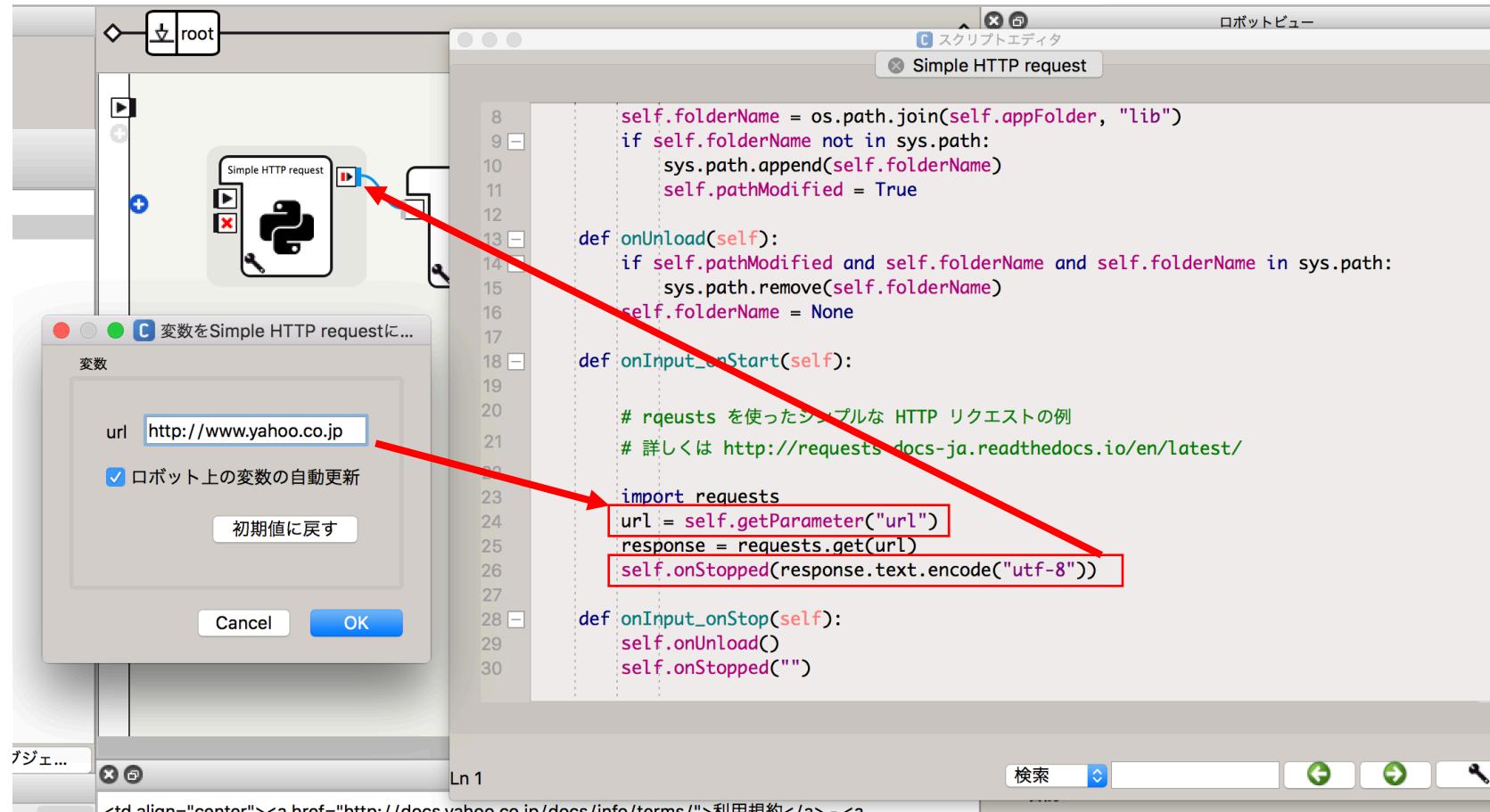
Simple HTTP request ボックス

1. プロパティー url に適当な URL を設定して、実行結果を確認。（「ログビューア」に結果が出る）



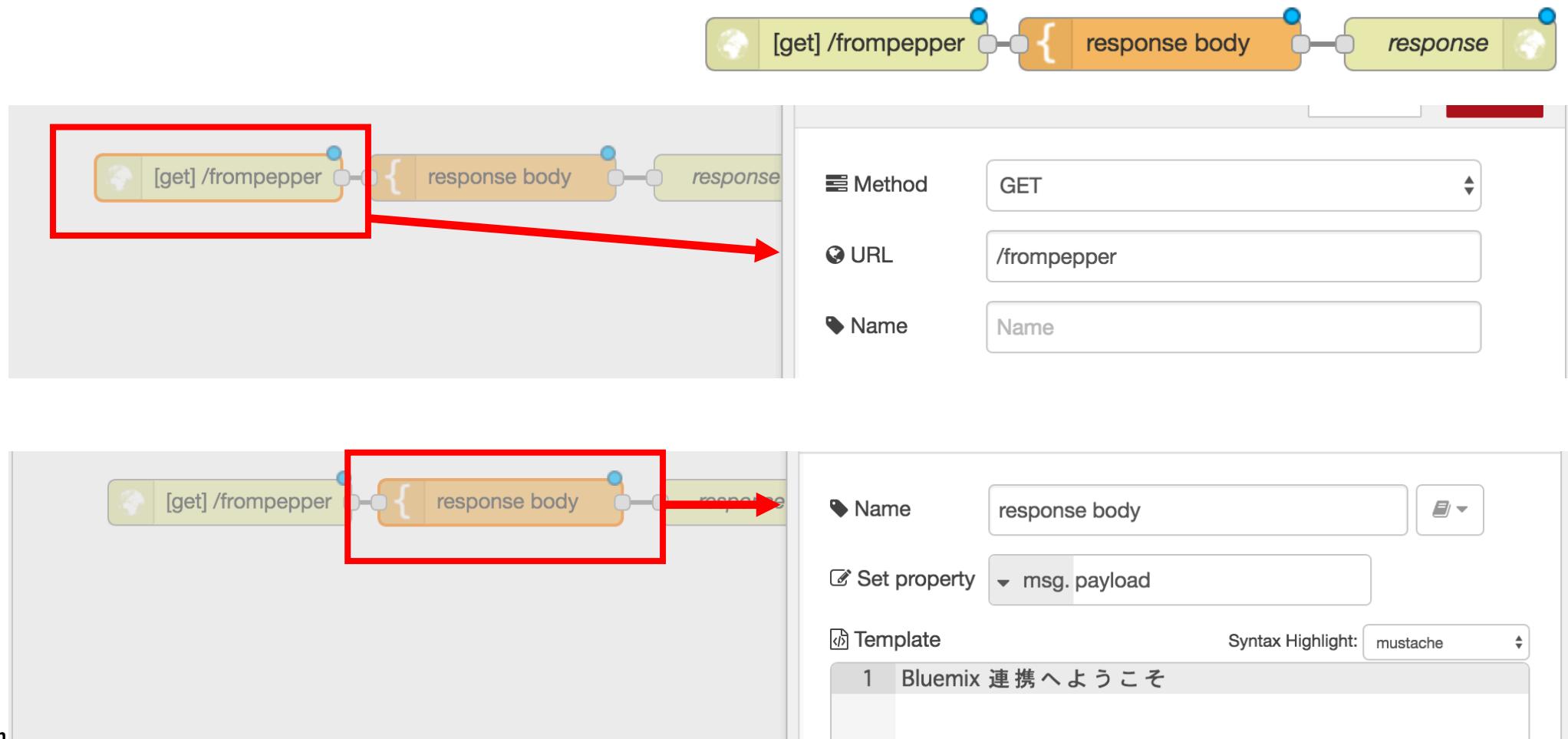
Simple HTTP request ボックス

2. スクリプトエディタでスクリプトを確認



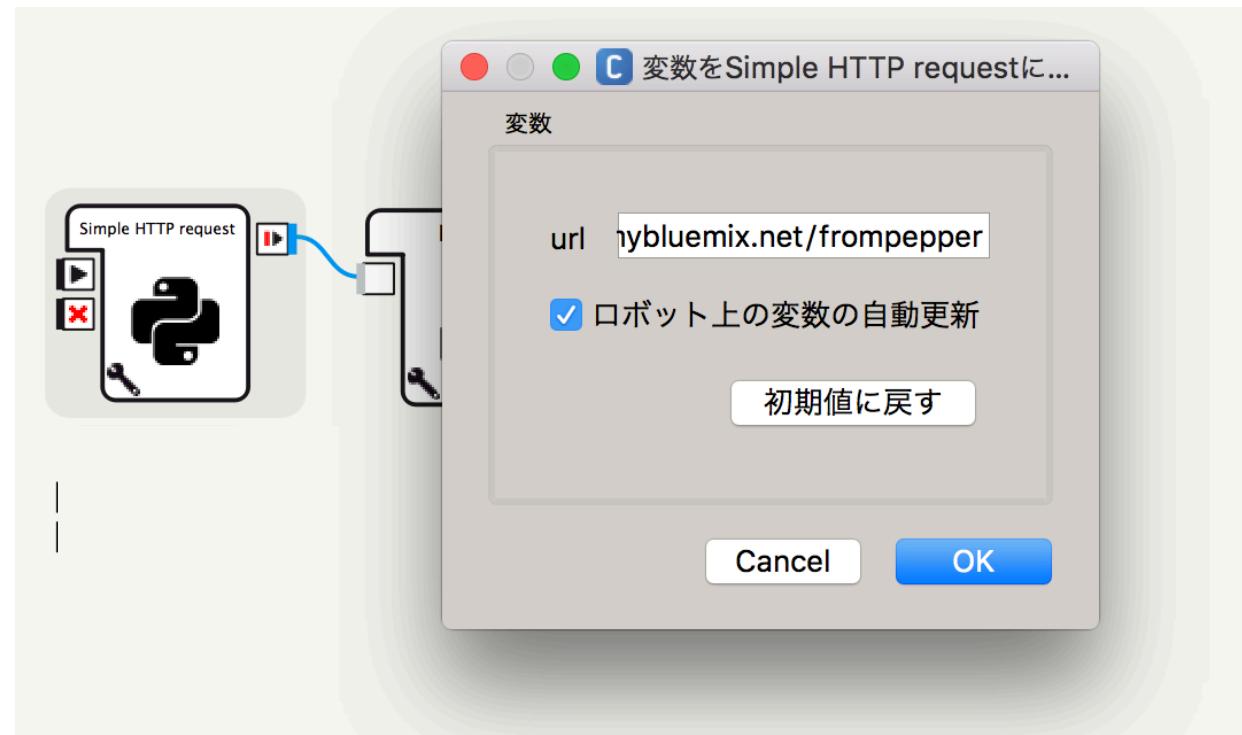
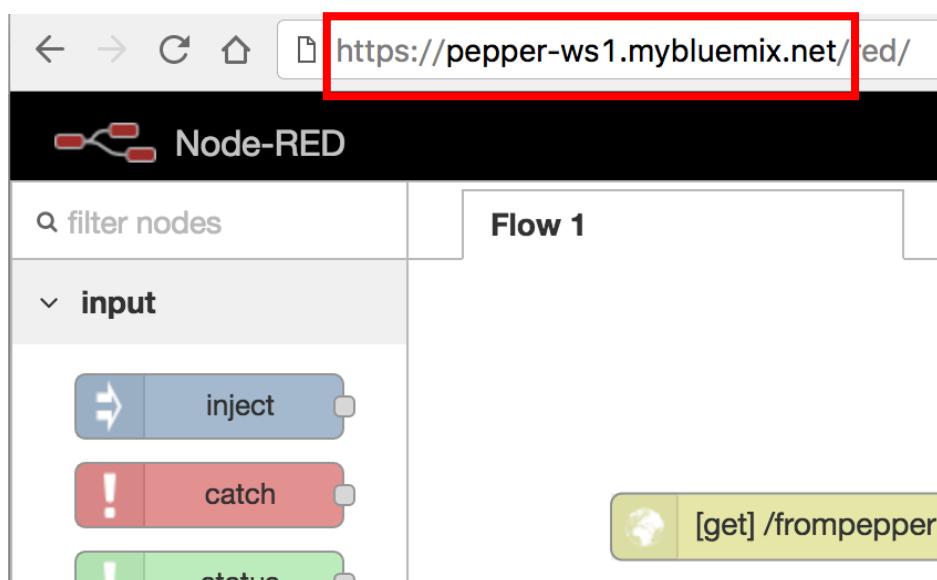
Simple HTTP request + Node-RED

1. Node-RED 側で HTTP リクエストの受け口作成



Simple HTTP request + Node-RED

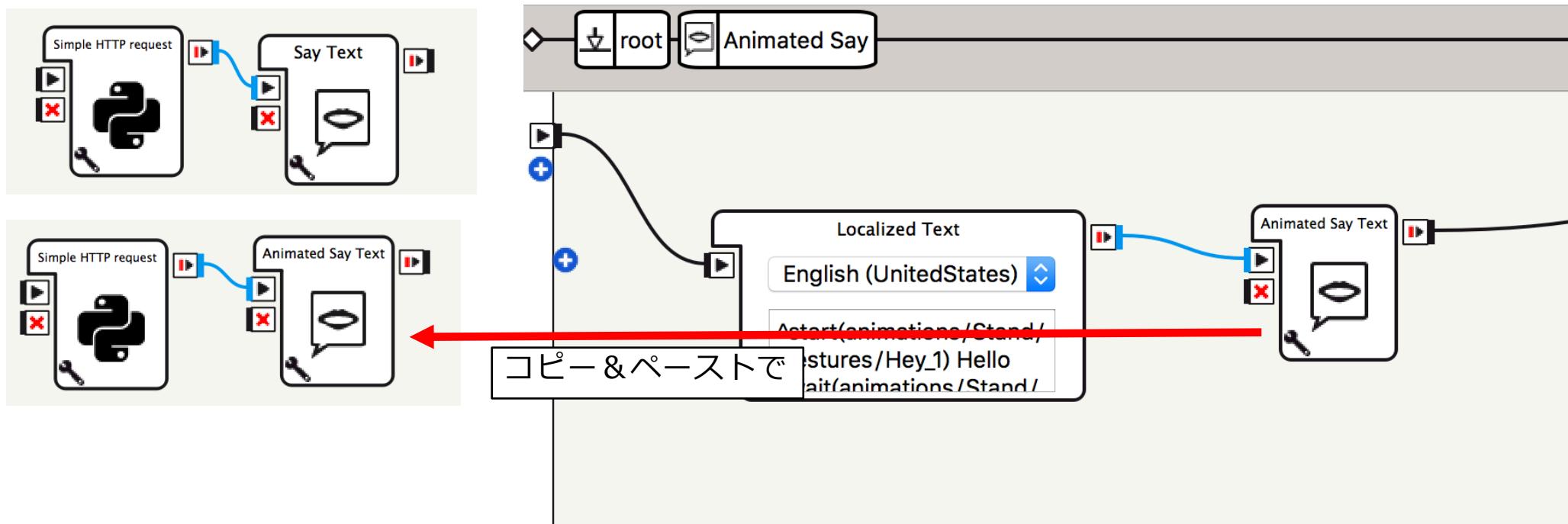
2. Pepper アプリプロジェクト、Simple HTTP Request 接続先を Node-RED の HTTP request ノードに



3. 実行して結果を確認

Pepper のクラウド連携 Tips.1 – Say Text

出力内容を Pepper に発話させるには Say Text ボックス（身振りなし）または Animated Say ボックのにある Animated Say Text（身振りあり）ボックスを使うのが楽

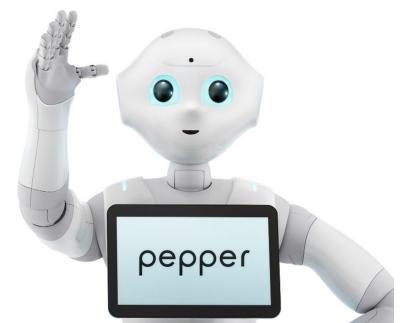


発話内容はバーチャルロボットでは「ダイアログ」ビューで確認できる

HTTPリクエスト(2) 画像データの送信

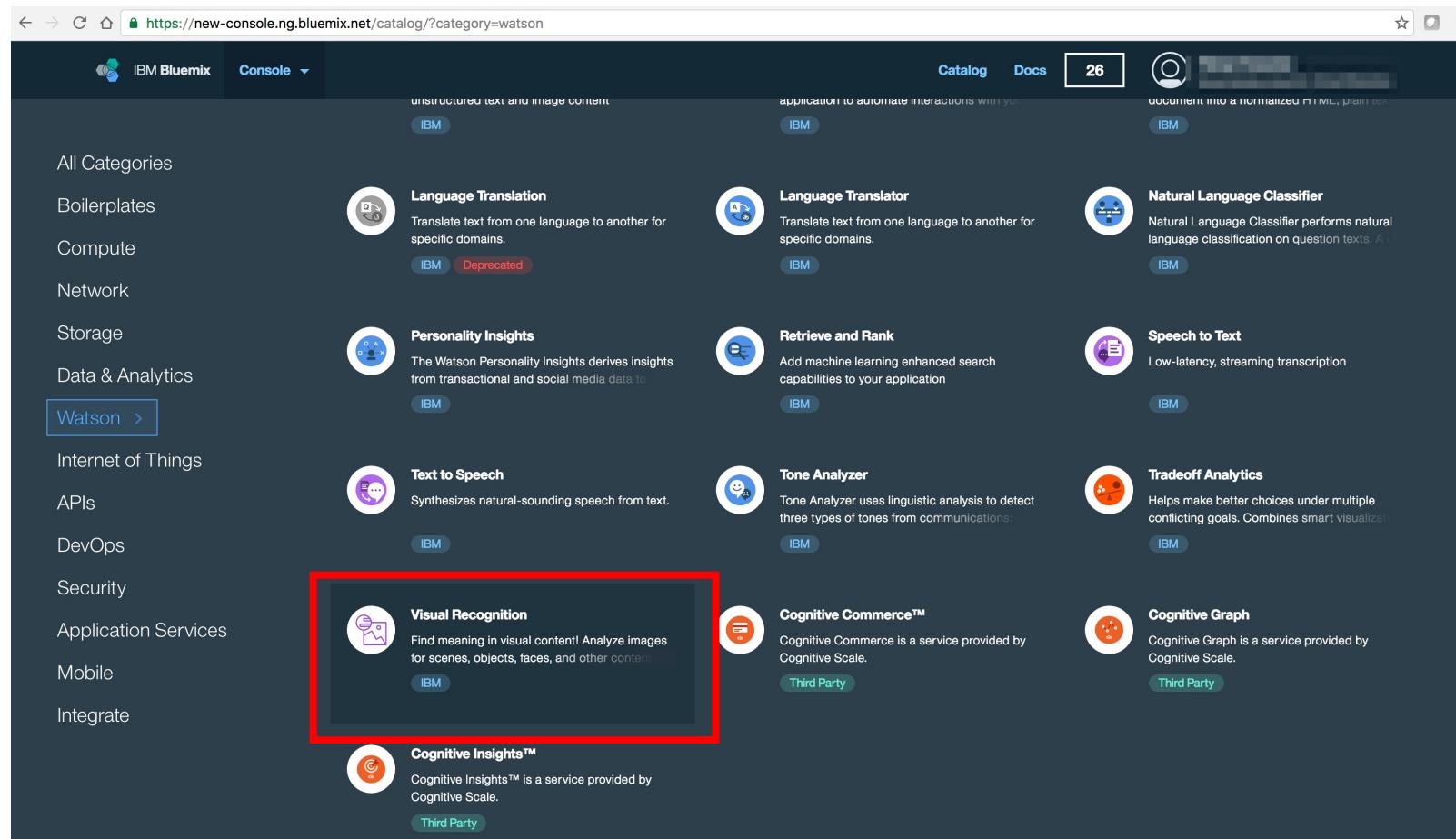
(Watson Visual Recognition で画像認識)

02uploadimage



Watson Visual Recognition で画像認識

1. IBM Bluemix コンソール、カタログ Watson の中の Visual Recognition を選択



Watson Visual Recognition で画像認識

2. Connect to で Node-RED アプリケーションを選択し [Create] ボタンでサービス作成

The screenshot shows the IBM Bluemix Catalog interface. At the top, there is a navigation bar with 'IBM Bluemix Catalog' on the left, a user icon, and 'Catalog Support Account' on the right. A '29' badge is in the top right corner. Below the navigation, there is a 'View All' link and a search bar.

The main content area is titled 'Visual Recognition'. It contains a brief description of the service: "Find meaning in visual content! Analyze images for scenes, objects, faces, and other content. Choose a default model off the shelf, or create your own custom classifier. Find similar images within a collection. Develop smart applications that analyze the visual content of images or video frames to understand what is happening in a scene." Below this is a 'Service name:' input field containing 'Visual Recognition-js'.

A red box highlights the 'Connect to:' dropdown menu, which is currently set to 'Node-RED'. Below this is a 'View Docs' link.

The 'Features' section lists several capabilities:

- General Classification: Generate class keywords that describe the image. Use your own images, or extract relevant image URLs from publicly accessible webpages for analysis.
- Visual Training: Create custom, unique visual classifiers. Use the service to recognize custom visual concepts that are not available with general classification.
- Face Detection: Detect human faces in the image. This service also provides a general indication of age range and gender of faces.
- Similar Image Search (BETA): Upload and search through image collections to find visually similar images.

The 'Pricing Plans' section shows a single plan:

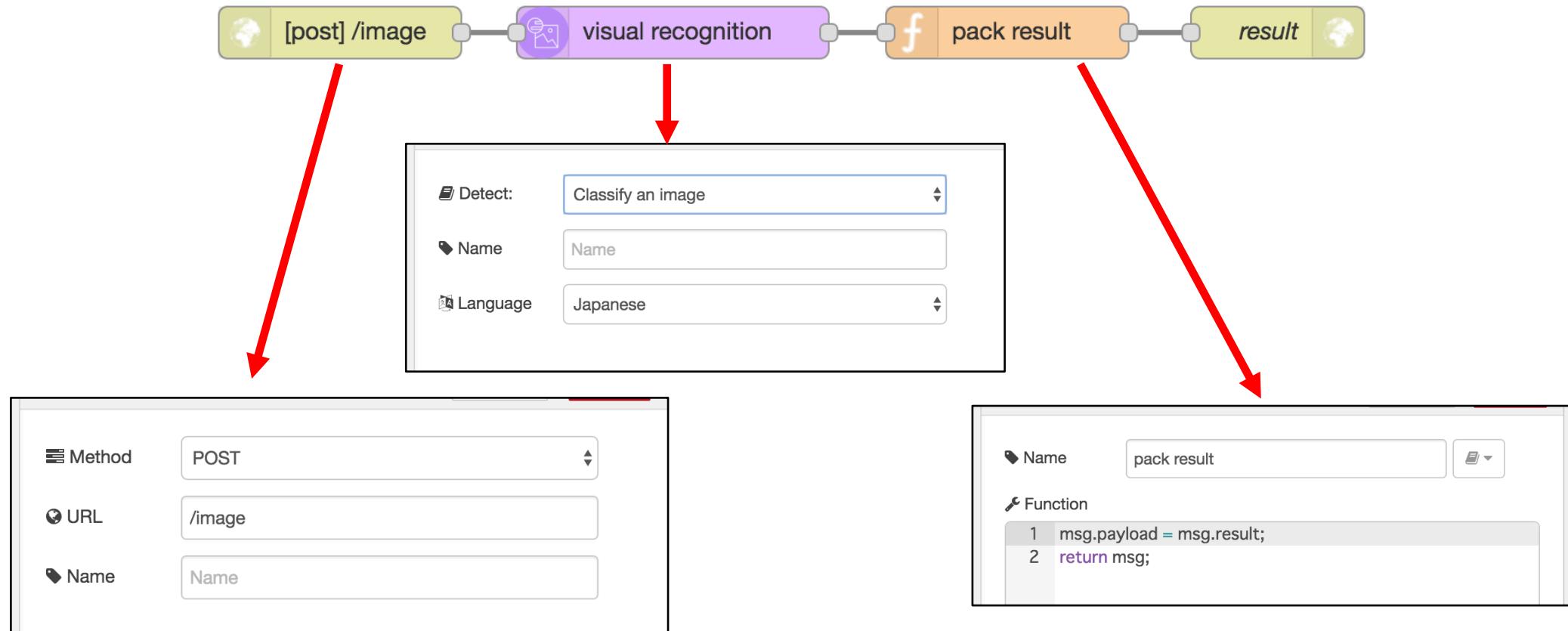
PLAN	FEATURES	PRICING
✓ Free	250 Events (images) per day per Bluemix Account 1 Instance per Bluemix Organization	Free

At the bottom, there are links for 'Need Help? Contact Bluemix Sales' and 'Estimate Monthly Cost Cost Calculator'. A red box highlights the 'Create' button at the bottom right.

3. アプリケーションをリストアージして、サービスを有効にする

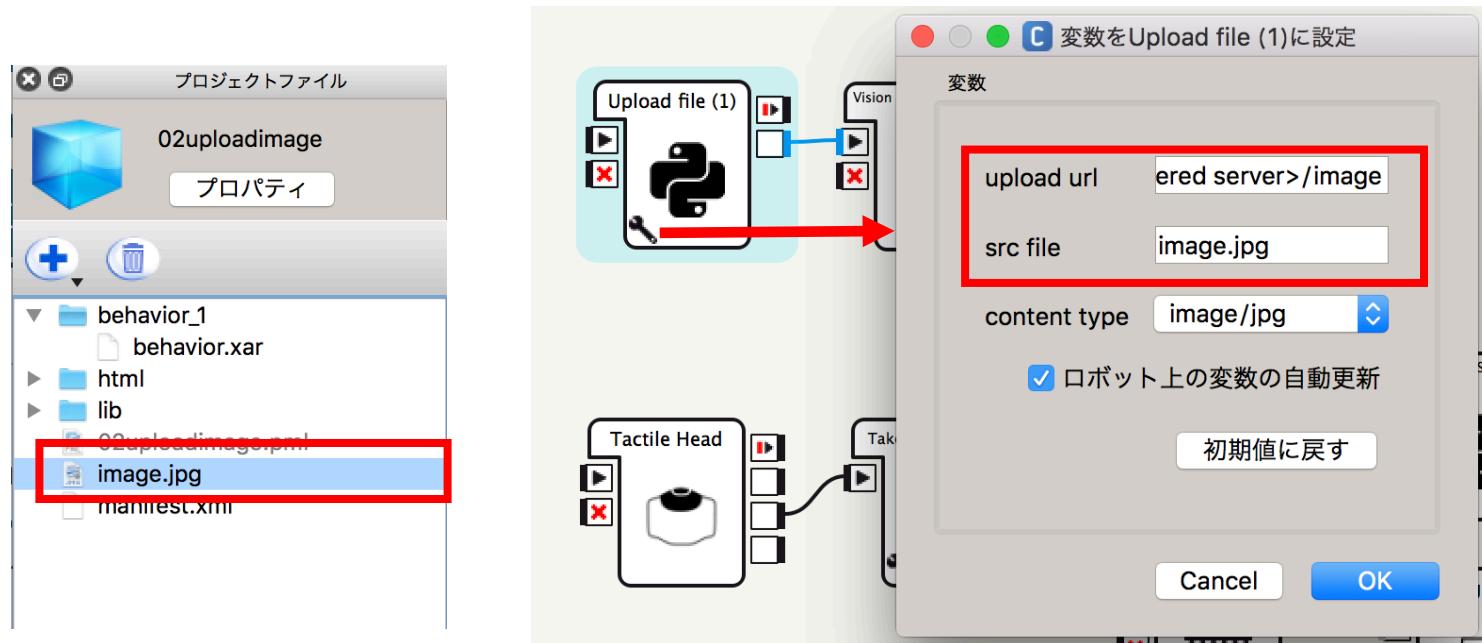
Node-RED で Pepper とのインターフェースを作成

1. Node-RED 側で HTTP リクエストで受け口、Visual Recognition につなぎ結果をレスポンスで返す



Pepper側 - Update file ボックスで画像ファイルを送る

1. 認識させる画像をプロジェクトに取り込む

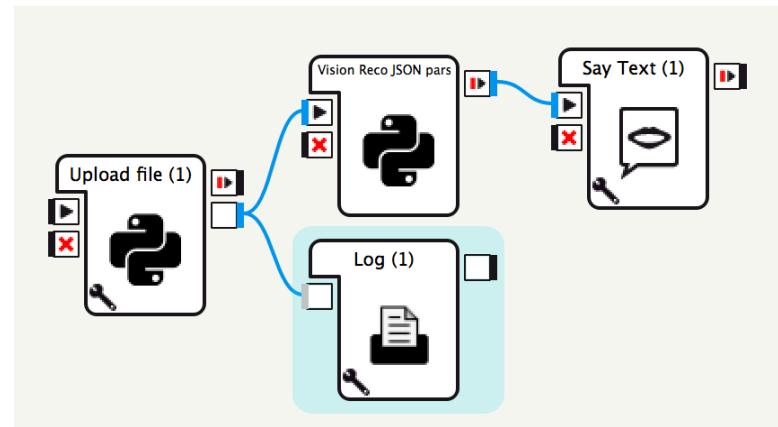


2. Upload file ボックス、'upload url' に Node-RED のアクセスポイント、'src file' に プロジェクトに保存した イメージファイル、'content-type' に image/jpg を設定

Pepper側 – 動作確認用の接続

3. Upload file ボックスは結果を ‘output’ 出力に出力する

これを Log ボックスにつなげてることで出力内容をログビューアで見る事ができる



ログビューア

```
test0.mybluemix.net/image?  
[INFO ] behavior.box :onInput_message:27  
_Behavior_lastUploadedChoregrapheBehaviorbehavior_1140469006410160:/Log_3: Message text:  
{"custom_classes":0,"images":[{"classifiers":[{"classes":[{"class":"植物","score":0.999997},{"class":"花","score":0.710949}],"classifier_id":"default","name":"default"}],"image":"11694-29-rsw3uq.jpg"}],"images_processed":1}
```

すべてのログを表示 ログレベル : 情報

This screenshot shows the 'Log Viewer' window. The title bar says 'ログビューア'. The main area displays the log message: 'test0.mybluemix.net/image? [INFO] behavior.box :onInput_message:27 _Behavior_lastUploadedChoregrapheBehaviorbehavior_1140469006410160:/Log_3: Message text: {"custom_classes":0,"images":[{"classifiers":[{"classes":[{"class":"植物","score":0.999997},{"class":"花","score":0.710949}],"classifier_id":"default","name":"default"}],"image":"11694-29-rsw3uq.jpg"}],"images_processed":1}'. At the bottom, there is a checkbox labeled 'すべてのログを表示' and a dropdown menu labeled 'ログレベル : 情報'.

Pepper のクラウド連携 Tips.2 – JSON モジュール

Watson Vision Recognition は結果を JSON 形式で返します。
JSON 形式のデータのやり取りには json モジュールの利用が便利です。

json.loads(JSON形式文字列) JSON 形式の文字列を python のオブジェクト
(ディクショナリオブジェクト) に変換

json.dumps(ディクショナリオブジェクト) python のオブジェクト
(ディクショナリオブジェクト) をJSON形式の
文字列に変換

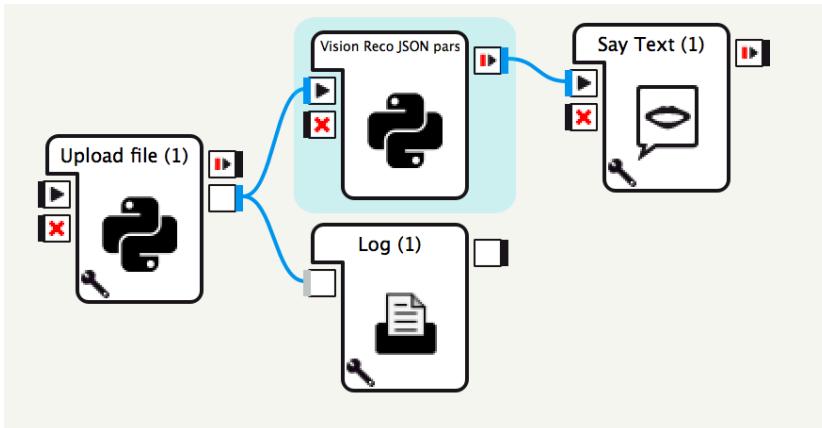
※ json.loads でデコードされたディクショナリオブジェクトの中の文字列は unicode 文字列、
一方 Pepper は文字列を str 型で扱う必要があるため、文字列のエンコードが必要
例： json 文字列 jsonStr から result の内容を取り出す

```
jsonObj = json.loads(jsonStr)
result = jsonObj["result"].encode("utf-8")
```

JSON モジュールを用いた結果データのパース処理例

1. Vision Reco JSON parts ボックス

Watson Visual Recognition からの結果をパース



```
def onInput.onStart(self, p):
    import json
    topScore = 0
    topResult = ""
    try:
        # IBM Watson Visual Recognition の仕様を確認して、レスポンスを解析
        # 仕様:
        # http://www.ibm.com/watson/developercloud/visual-recognition/api/v3/?python#classify_an_image
        #
        jsonResult = json.loads(p)
        classifiers = jsonResult["images"][0]["classifiers"]
        for aClassifier in classifiers:
            classes = aClassifier["classes"]
            for aClass in classes:
                score = aClass["score"]
                result = aClass["class"].encode("utf-8")
                self.logger.info("result:" + result + " score:" + str(score))
                if topScore < score:
                    topScore = score
                    topResult = result

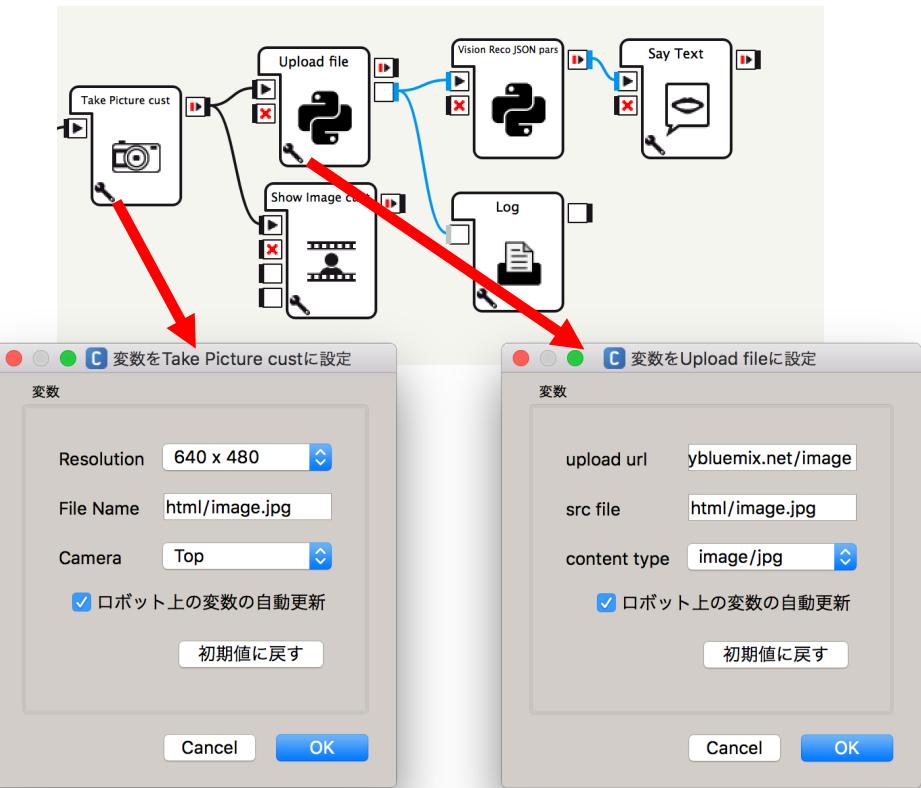
    except:
        self.logger.info("Parsing JSON response failed")

    if topScore > 0:
        response = "%d パーセントの確率で %s です" % (int(topScore * 100), topResult)
    else:
        response = "わかりませんでした"
```

Pepper が撮った写真を画像認識

※この実装を動作確認するためには実機での実行が必要です

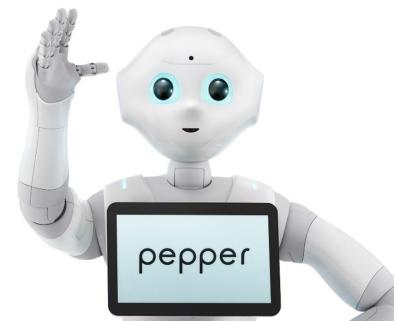
- 'Take Picture cust' ボックス
('Take Picture cust' ボックスは Take Picture ボックスをベースに撮った写真をプロジェクトフォルダー内に保存するようにカスタマイズしたボックスです)
- プロジェクトに html フォルダーを作りそこに写真を保存するようすれば、タブレット表示と画像認識を同じ写真で行うことが可能です



HTTPリクエスト(3) 音声データの送信

(Watson Speech To Text で音声認識)

03uploadaudio



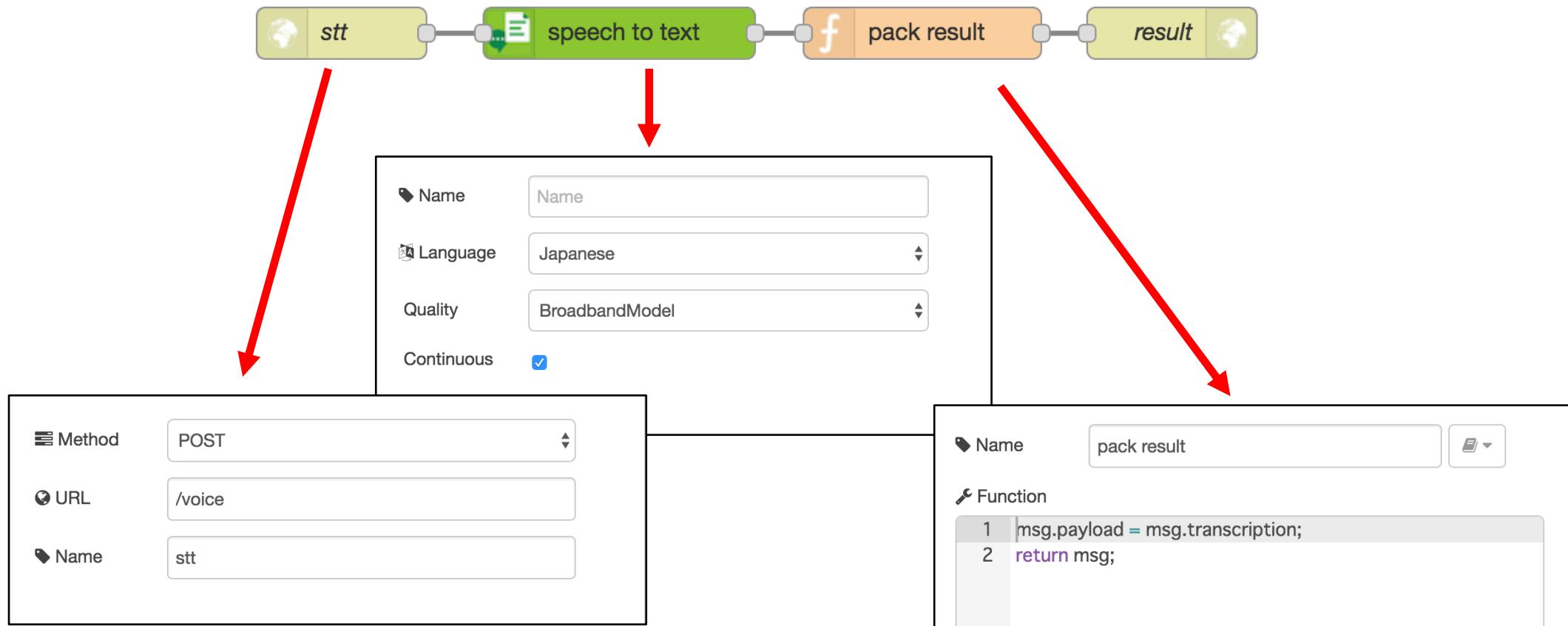
Speech to Text で音声認識

1. IBM Bluemix コンソール、カタログ Watson の中の Speech To Text を選択、アプリに追加

The screenshot shows the IBM Bluemix Catalog interface. On the left, there's a sidebar with categories like All Categories, Boilerplates, Compute, Network, Storage, Data & Analytics, Watson (which is selected and highlighted with a blue border), Internet of Things, APIs, DevOps, Security, Application Services, Mobile, and Integrate. The main area displays various Watson services in a grid. The 'Speech to Text' service is highlighted with a red box. Other visible services include Language Translation, Language Translator, Natural Language Classifier, Personality Insights, Retrieve and Rank, Tone Analyzer, Text to Speech, Visual Recognition, Cognitive Commerce™, Cognitive Graph, and Cognitive Insights™.

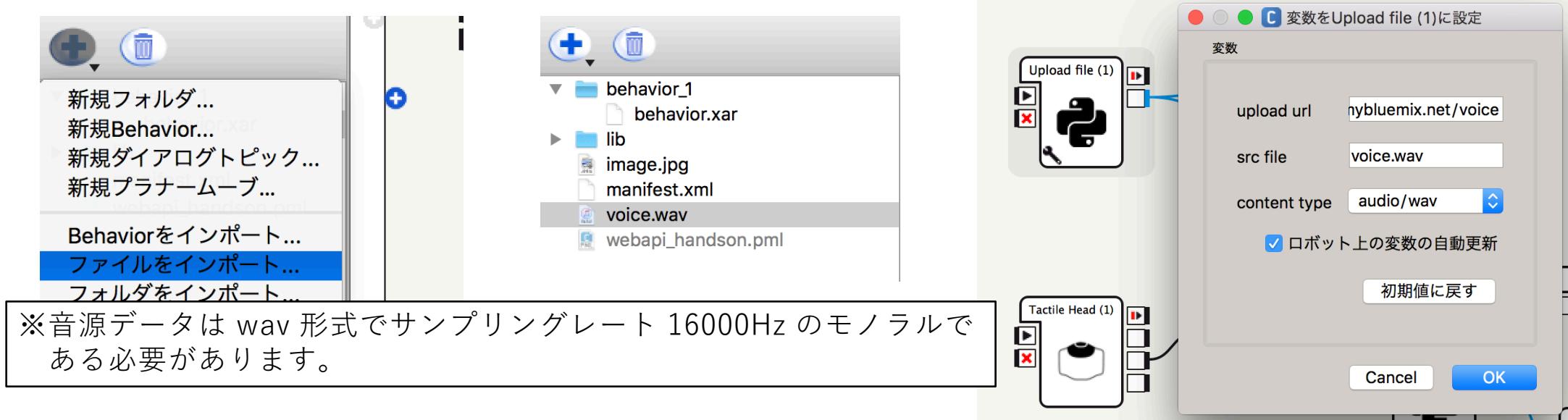
Node-RED で Pepper とのインターフェースを作成

1. Node-RED 側で HTTP リクエストで受け口、Speech to Text につなぎ結果をレスポンスで返す



Pepper側 - Update ファイルボックスで音声ファイルを送る

1. 認識させる音声データをプロジェクトに取り込む



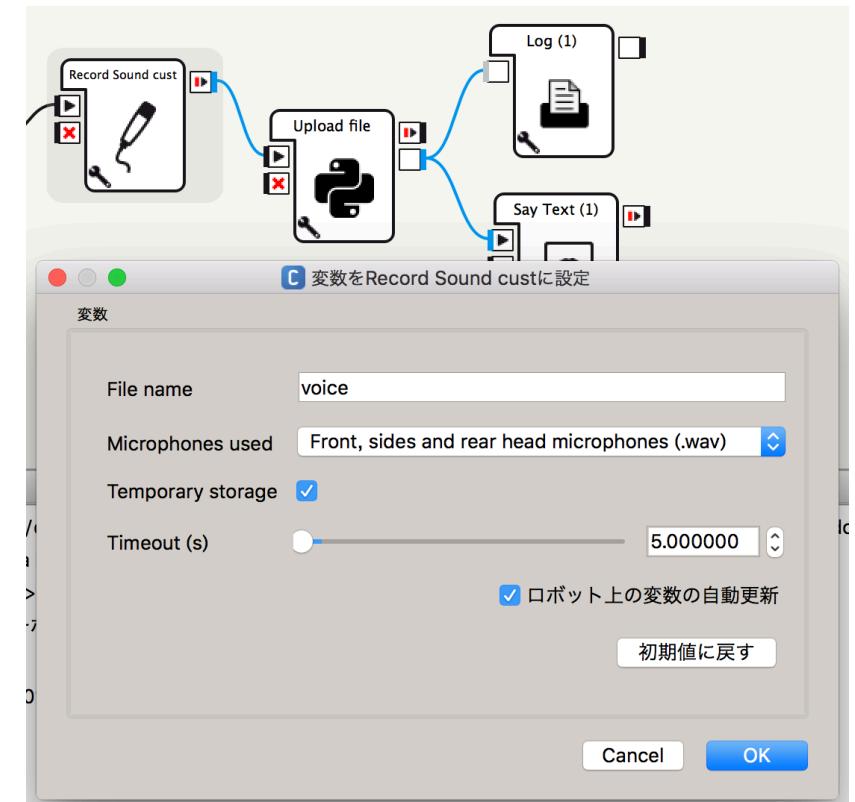
※音源データは wav 形式でサンプリングレート 16000Hz のモノラルである必要があります。

2. 'Upload file' ボックス、upload url に Node-RED のアクセスポイント、src file にプロジェクトに保存した 音源ファイル、'content-type' に audio/wav を設定
3. 'Say To Text' ボックス、または 'Log' ボックスにつなげて結果を確認

Pepper が録音した音声を音声認識

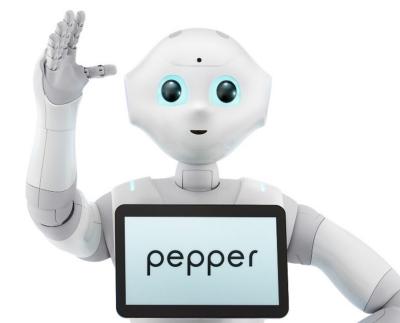
※この実装を動作確認するためには実機での実行が必要です

- ‘Record Sound cust’ ボックス
('Record Sound cust' ボックスは
‘Record Sound’ ボックスをベースに
録音した音声をプロジェクトフォルダ
内に保存するようにカスタマイズ
したボックスです)
- 保存するファイル名をプロパティで
設定（拡張子はつけません）



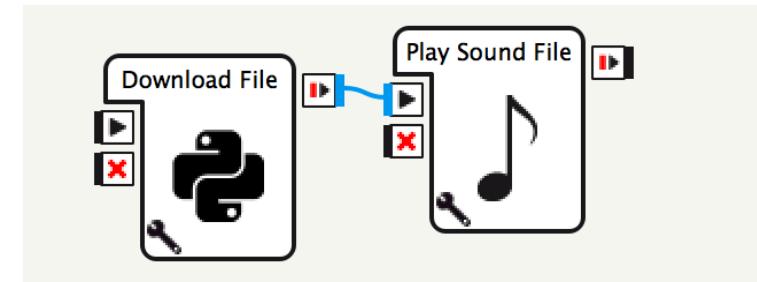
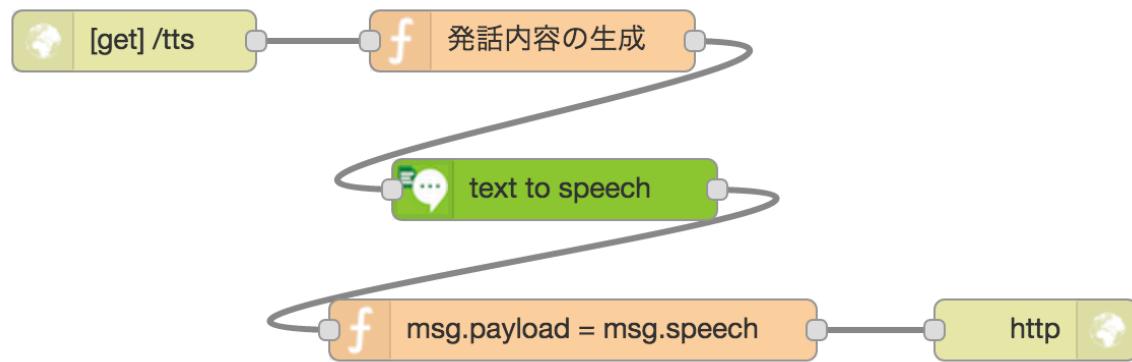
HTTPリクエスト(4) データの保存

04downloadfile



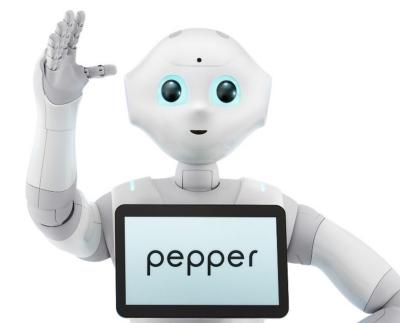
Download file ボックスを使ってHTTPレスポンスをファイルに保存

- 例えは Watson Text to Speech が音声合成した音声データを Pepper 本体に保存、再生



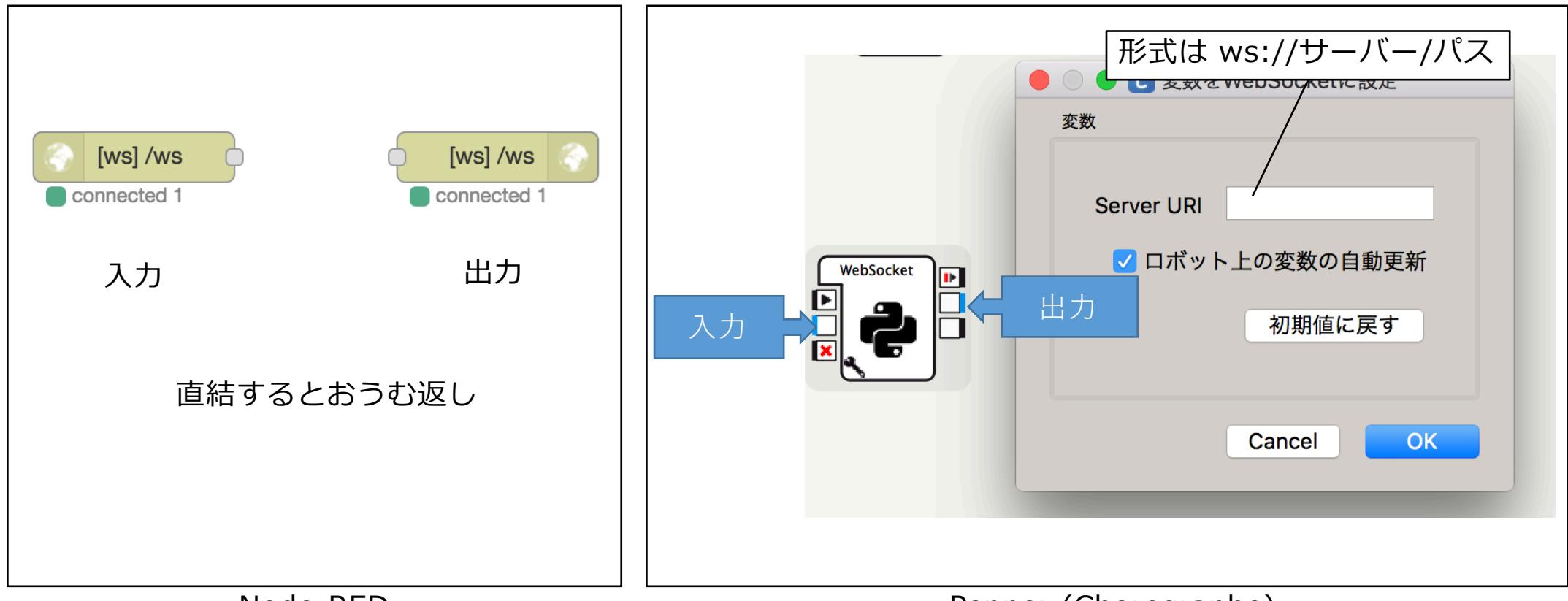
WebSocket 通信

05websocket



WebSocket ボックスを使って Node-RED と通信

WebSocket を使って、双方向、リアルタイムの通信を実現



Node-RED

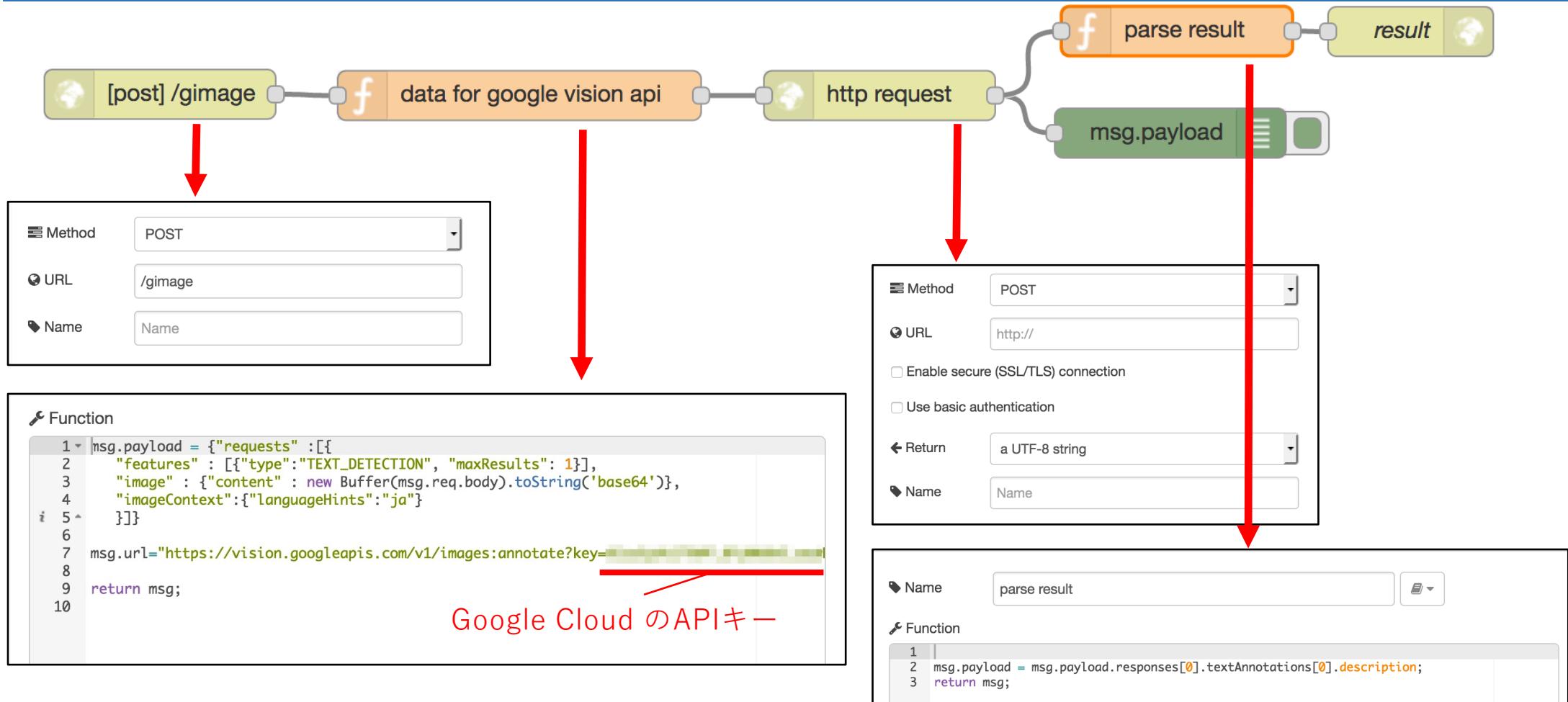
Pepper (Choregraphe)

外部サービスを Node-RED 経由で利用

06useotherservice



Node-RED 経由で Google Cloud Vision API を呼んでみる

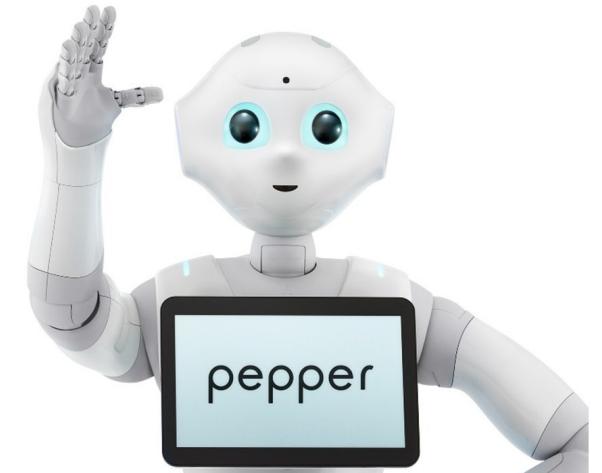


※ : Vision API の有効化、API キーの設定は <https://console.developers.google.com/> から

IBM Bluemix と Pepper の連携

手法	連携方法	サービスを利用するための認証	サービスを利用するロジック	適用場面
各サービスの API に直接アクセス	Web API にアクセス	Pepper が行う	Pepper 側で実装	ミニマムのシステム構成でシステムを構築したい場合。API の機能を最大限利用したい場合
Node-RED を使う		Node-RED が行う	Node-RED 側と Pepper 側で分散可能	ラピッドプロトタイピングに適したソリューション
サービス提供の SDK を使う	サービス提供の SDK をプロジェクトに取り込む	Pepper が行う (SDK を使って)	Pepper 側で実装	<ul style="list-style-type: none">Watson Developer Cloud Python SDKBluemix 全体に関する Python での SDK 提供はない。

各サービスの API に
直接アクセス



各サービスの API に直接アクセス利点と欠点

- 利点
 - 各サービスAPI に直接アクセスすることで機能を最大限利用できる
 - 最小限のシステム構成でシステム構築
- 欠点
 - リクエスト、レスポンスのデータ整形を Pepper で行う必要あり。
 - サービスを利用するための認証情報を Pepper 側で保持しなければならない
 - 安全に、セキュアに認証情報を保存する仕組みが必要

例：Watson Speech to Text の WebSocket インターフェースを使ってライブSTT

- Speech to Text のサービスクレデンシャルを作成

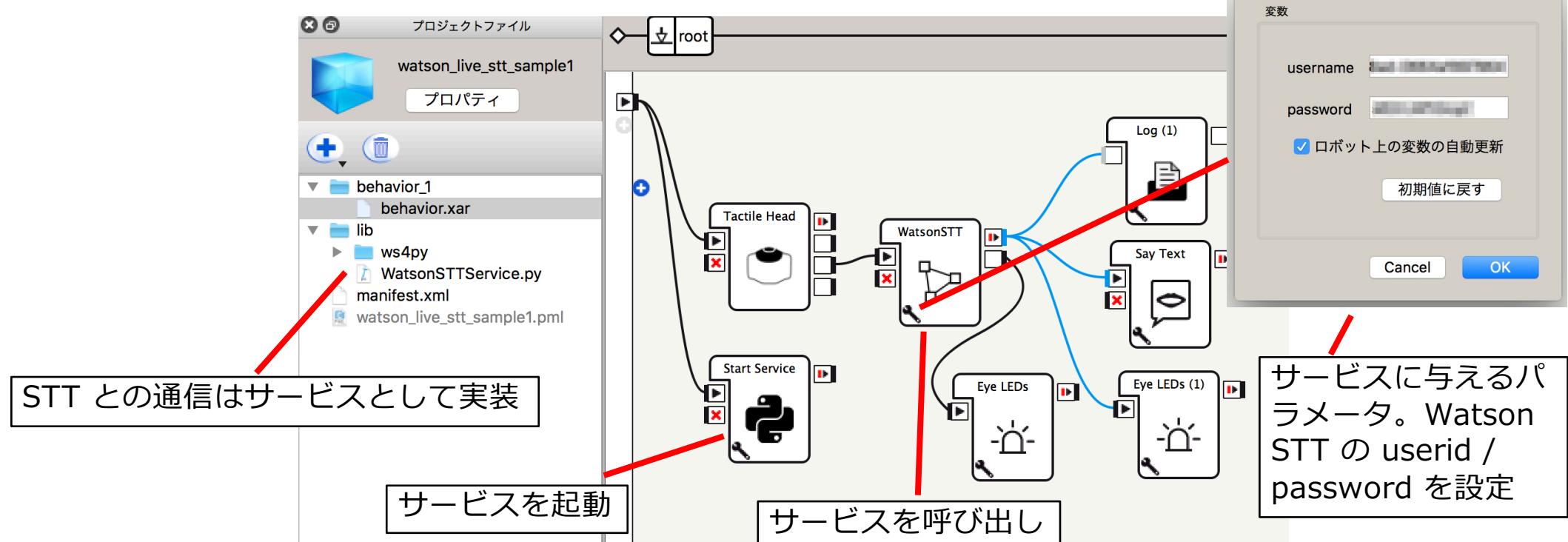
The screenshot shows the 'Service Credentials' section of the Watson service in the IBM Bluemix interface. A red box highlights the 'Service Credentials' tab under 'Manage'. Another red box highlights the 'New Credential +' button. A third red box highlights the JSON credentials for 'Credentials-1', which include the URL, password, and username.

KEY NAME	DATE CREATED	ACTIONS
Credentials-1	Oct 30, 2016 - 06:41:17	View Credentials Delete

```
{  
  "url": "https://stream.watsonplatform.net/speech-to-text/api",  
  "password": "REDACTED",  
  "username": "REDACTED"  
}
```

例：Watson Speech to Text の WebSocket インターフェースを使ってライブSTT

- Pepper アプリサンプル `watson_live_stt_sample1`
 - Watson STT が認識した音声をおうむ返しで返します。



※サービスの実装は次の Watson の記事を参考にしました

<https://www.ibm.com/blogs/watson/2016/07/getting-robots-listen-using-watsons-speech-text-service/>

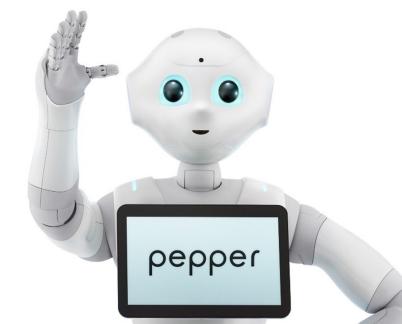
IBM Bluemix と Pepper の連携

手法	連携方法	サービスを利用するための認証	サービスを利用するロジック	適用場面
各サービスの API に直接アクセス	Web API にアクセス	Pepper が行う	Pepper 側で実装	ミニマムのシステム構成でシステムを構築したい場合・API の機能を最大限利用したい場合
Node-RED を使う		Node-RED が行う	Node-RED 側と Pepper 側で分散可能	ラピッドプロトタイピングに適したソリューション
サービス提供の SDK を使う	サービス提供の SDK をプロジェクトに取り込む	Pepper が行う (SDK を使って)	Pepper 側で実装	<ul style="list-style-type: none">Watson Developer Cloud Python SDKBluemix 全体に関する Python での SDK 提供はない。

サービス提供の SDK を使う



外部モジュールの取り込み方

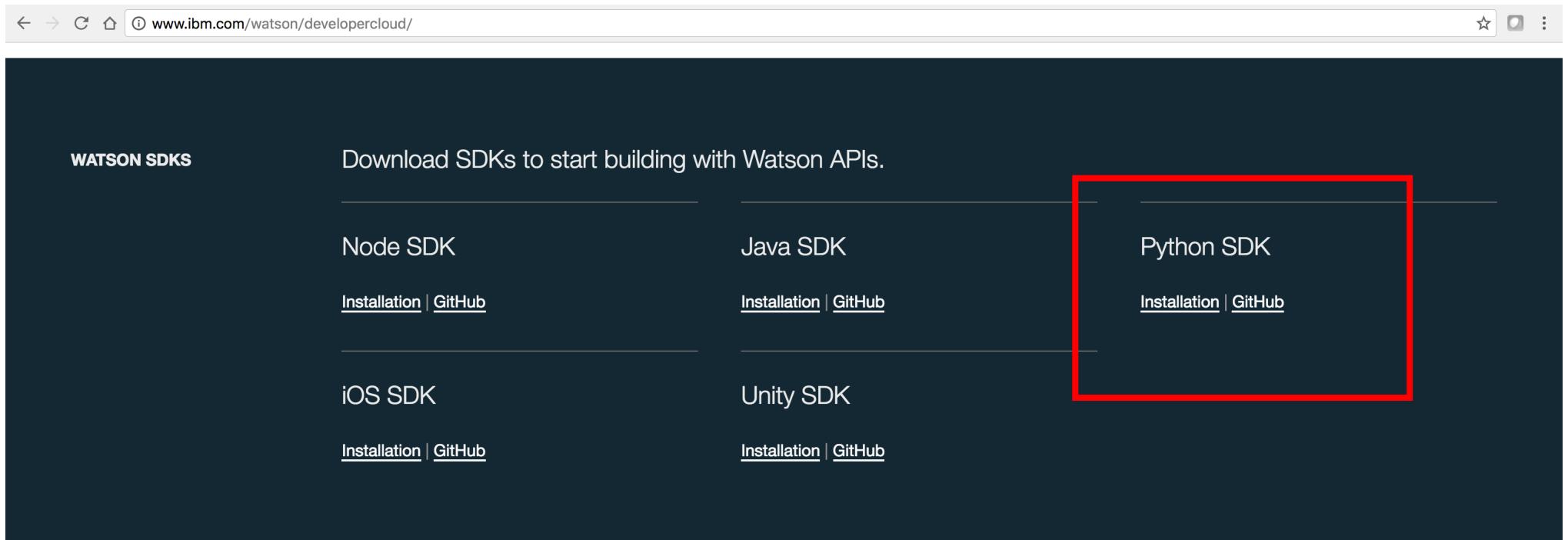


外部モジュールの取り込み

- 方針：
 - プロジェクトに任意のフォルダを作成（例：lib）
 - フォルダに外部モジュールのファイルを保存
 - プロジェクト実行時に python のモジュール参照パスに任意のフォルダを追加
- 注意点：
 - アプリの実行は Pepper 本体で行われる。 Pepper 本体は Intel Atom CPU 上で動く Gentoo Linux。この環境で動くファイルが一通りフォルダ内にそろっている必要あり。特にネイティブコードを含むモジュールなど注意が必要
- 手法：
 - 手法 1：モジュールがアーカイブされているものはこれを入手
 - 手法 2：pip を使う場合はインストール先を指定、アウトプットを取り込む
 - `pip install --install-option="--prefix=$PREFIX_PATH" package_name`
 - 手法 3：NAOqi バーチャルマシンを使ってインストールイメージを作る

Watson Developer Cloud の python SDK を使う

- <http://www.ibm.com/watson/developercloud/>



サンプルプログラム

```
#put initialization code here
pass

def onUnload(self):
    #put clean-up code here
    pass

def onInput_onStart(self):
    import json
    from os.path import join, dirname
    from watson_developer_cloud import TextToSpeechV1

    appFolder = self.behaviorAbsolutePath().replace(self.behaviorRelativePath(), "")

    text_to_speech = TextToSpeechV1(
        username='USEID',
        password='PASSWORD',
        x_watson_learning_opt_out=True) # Optional flag

    self.logger.info(json.dumps(text_to_speech.voices(), indent=2))

    path = join(appFolder, 'output.wav')
    with open(path, 'wb') as audio_file:
        audio_file.write(text_to_speech.synthesize('Hello world!', accept='audio/wav', voice="en-US_AllisonVoice"))

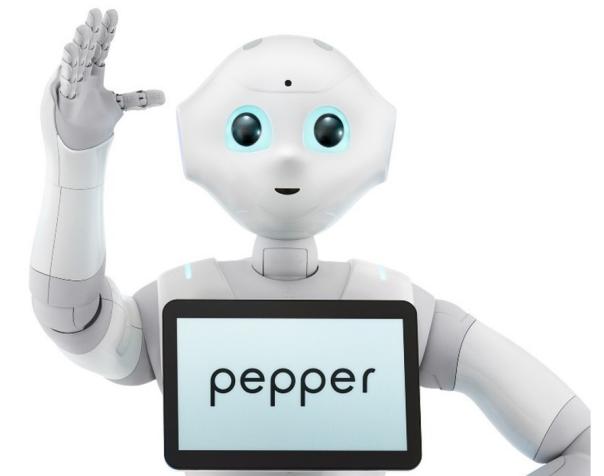
    self.logger.info(json.dumps(text_to_speech.pronunciation('Watson', pronunciation_format='spr'), indent=2))

    self.logger.info(json.dumps(text_to_speech.customizations(), indent=2))

    self.onStopped(path)

def onInput_onStop(self):
    self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
    self.onStopped("") #activate the output of the box
```

Pepper のタブレット

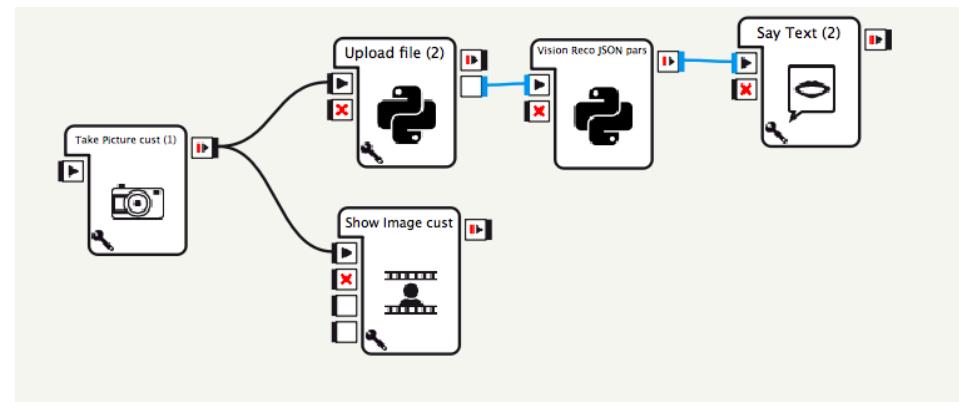


タブレットの活用

- 活用シーン：
 - 音声認識ではうまくコミュニケーションが取れない場合の補助
 - 既存 Web サービス画面の活用
- 機能：
 - 画像の表示
 - タッチ位置の検出
 - HTML ページの表示
 - ビデオファイルの再生（ただし音はタブレットからしか出ない）
 - プロジェクトのファイルをタブレットに表示させることも、外部のリソースを開くことも可能

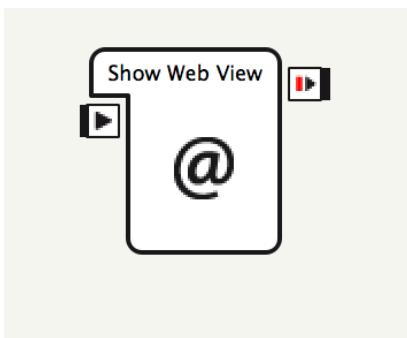
例：プロジェクトのファイルを開く

- ・プロジェクトの中のHTMLフォルダー内のファイルを開くことができる
- ・例えばアプリが撮った写真をタブレットに表示したい場合、プロジェクトの HTML フォルダー内に保存すればいい。
- ・キャッシュに注意
 - ・URL の末尾に ?乱数 でキャッシュブレーク
 - ・イメージの表示は ALTabletService.showImage() の代わりに ALTabletService.showImageNoCache() を使う



例：外部ファイルを開く

- HTML ページについては標準ではボックスがないので標準ボックスをカスタマイズして
- 例えば Show Web View ボックス、スクリプトで showWebView を呼び出しているところの引数に URL を与える



```
21
22 def onInput_onStart(self):
23     # We create TabletService here in order to avoid
24     # problems with connections and disconnections of the tablet during
25     # the life of the application
26     tabletService = self._getTabletService()
27     if tabletService:
28         tabletService.showWebview("http://www.yahoo.co.jp") # <== 変更
29     else:
30         self.logger.warning("ALTTabletService not found.")  
self.onStopped()
```

終わり

