



## بناء التدفقات عبر البروتوكولات

في هذا الدرس سوف نتطلع لرؤية المزيد من العقد الأكثر شيوعا و بناء مشاريع باستخدام بعض العقد التي تم شرحها في الدروس السابقة . سوف نبدأ بسلسلة من الأمثلة القائمة حول بروتوكول MQTT و التي تبين كيفية ربط مجموعة من عقد معالج الرسالة . ثم سيتم النظر لطرق أخرى للحصول على الرسائل من و إلى التدفقات باستخدام بروتوكولات مثل TCP و UDP و مآخذ الويبsockets .

في نهاية هذه المحاضرة سيكون لديك فهم أفضل لبعض العقد الأساسية المستخدمة في تدفق Node-RED . وسيكون لديك القدرة على بناء التدفقات المعالجة المتطورة التي تأخذ الأحداث في العالم الحقيقي و معالجتها و توليد استجابات لإيصال النتائج خارج تدفق خاص بك .

السلسلة التالية من الأمثلة تبنى على عقدة mqtt ، الذي يوفر وسيلة ملائمة لإتخاذ المدخلات من وسيط MQTT . MQTT هو مثال على نظام نشر / اشتراك والذي يتيح لأجهزة الاستشعار بنشر التحديثات التي تسلمها إلى العميل المشترك في ذلك المستشعر . يستخدم MQTT نموذج الموضوع للسماح للناشرين (مثل أجهزة الاستشعار) لإنشاء الموضوعات و نشر البيانات للمواضيع ويمكن للآخرين الاشتراك في هذا الموضوع للحصول على اشعارات غير متزامنة من البيانات المرسله لهذا الموضوع .

أنظمة pub/Sub هي طريقة رائعة لربط النظم الموزعة المتباعدة وهم خريطة جيدة لنموذج أنماط IOT حيث الأجهزة أو الأشياء تولد الأحداث التي ترغب في مشاركتها. بروتوكول MQTT ، فضلا عن كونها غير متزامنة هو أيضا خفيفة الوزن .

## تلقي JSON عبر رسالة MQTT

لإستخدام عقدة MQTT ، تحتاج الوصول إلى الوسيط broker. هناك عدد من الخوادم تشغيل MQTT مجانية ، على سبيل المثال <http://test.mosquitto.org/> ، او تلك التي سيتم استخدامها في هذا الدرس [www.hivemq.com](http://www.hivemq.com) .

باستخدام عنوان الوسيط و الموضوع، يمكنك تكوين/إعداد عقدة المدخلات mqtt للإشتراك في هذا الموضوع . الأمر الذي أدى إلى إنشاء رسالة جديدة هو نشر البيانات الجديدة حول هذا الموضوع . و تحتوي الرسالة على معلومات عن البيانات المنشورة، وتتضمن البيانات نفسها في msg.payload و موضوع وسيط MQTT في msg.topic .

للبدء من عقدة mqtt ، عليك استخدام وسيط mqtt المجاني (hivemq) وهو متاح عبر <http://www.hivemq.com/showcase/public-mqtt-broker/> . بالطبع يمكنك استخدام أي وسيط MQTT .

أولا، قم بسحب و إسقاط عقدة المدخلات mqtt إعدادها للوسيط. لا تنسى تكوين/إعداد الموضوع ليكون فريد من نوعه (unique) في هذا المثال نحن نستخدم noderedlecture/sensor ولكن عليك استخدام الموضوع الفريد الخاص بك ، مثال : your> name>/sensor .

Flow 3

mqtt in > Add new mqtt-broker config node

Cancel Add

Connection Security Birth Message Will Message

Server broker.mqtt-dashboard.com Port 1883


☐ Enable secure (SSL/TLS) connection

Client ID noderedlecture/sensor

Keep alive time (s) 60 ☒ Use clean session

☒ Use legacy MQTT 3.1 support

هناك العديد من الطرق التي يمكنك من إرسال رسائل mqtt إلى hivemq . ويمكنك استخدام مآخذها الويب websockets للتعامل (<http://www.hivemq.com/demos/websocket-client/>) ، ولوحة الإعدادات لـ mqtt (mqtt dashboard) (<http://www.mqtt-dashboard.com/dashboard>) أو مكتبك الخاصة . عليك استخدام مآخذها الويب للتعامل websocket client في هذا المثال ، لذلك انتقل إلى تلك الصفحة وقم بالاتصال إلى الوسيط . سوف تقوم بنشر سلسلة JSON المرتبطة بالموضوع الذي قمت بتكوينه/إعداده ولرؤية ذلك قم باستخدام عقدة qmtt وعقدة json .

 **HiveMQ** Websockets Client Showcase

Connection connected

Host broker.mqtdashboard.com Port 8000 ClientID clientid-0vSSg1enud Disconnect

Username Password Keep Alive 60 SSL ☐ Clean Session ☒

Last-Will Topic Last-Will QoS 0 Last-Will Retain ☐

Last-Will Message

Publish

Topic noderedlecture/sensor QoS 0 Retain ☐ Publish

Message {"analyze":false, "value":10}

Subscriptions Add New Topic Subscription

بمجرد أن تقوم بإرسال سلسلة JSON ، سوف تحتاج إلى تحليل الرسالة التي تولدها عقدة mqtt عندما تتلقى رسالة MQTT . للقيام بذلك، سوف تحتاج إلى سحب وإسقاط عقدة json و توصيلها مع نقطة الإخراج في عقدة mqtt .

عقدة json هي نوع من convenience function ، حيث أنه يوزع الرسالة الواردة ويحاول تحويلها من/إلى JSON . لذلك إذا قمت بإرسال سلسلة JSON سيقوم بتحويلها إلى جافاسكريبت JavaScript والعكس .

قم بتوصيل عقدة debug إلى عقدة json ثم قم بعملية النشر deploy ، استخدم لوحة إعدادات HiveMQ لإرسال سلسلة JSON كما هو موضح بالشكل أدناه :

The screenshot shows a Node-RED workspace with a flow consisting of three nodes: a purple 'noderedlecture/sensor' node (labeled 'connected'), a yellow 'json' node, and a green 'msg.payload' node. The flow is connected from the sensor node to the json node, and then to the msg.payload node. On the right, the debug console displays a message received at 13/08/2015 12:22:41 with ID [5639094c.a9c6f8]. The message is an object: { "analyze": false, "value": 10 }.

قم بالنظر عن كثب في المخرجات . يمكنك أن ترى msg.payload تحتوي على object و الذي يحتوي على حقلين حقل التحليل (analyze) و حقل القيمة (value) كما رأينا في الدرس السابق يمكنك الوصول إلى هذه الحقول عبر msg.payload.analyze و msg.payload.value . دعونا نلقي نظرة على العقدة وما يمكنها أن تفعل .

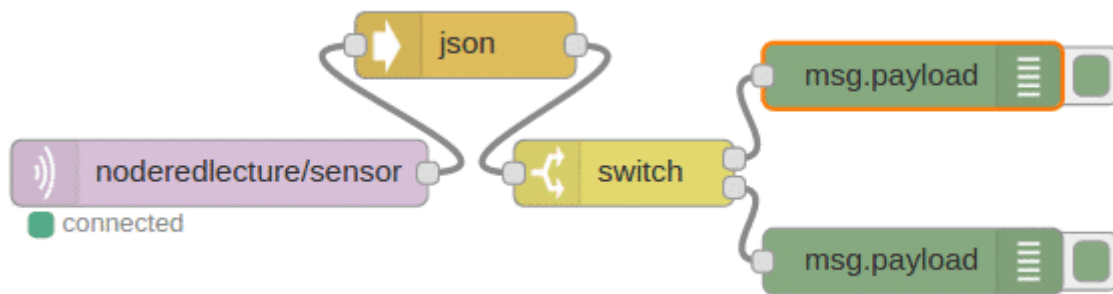
## استخدام عقدة التبديل لتوجيه الرسالة اعتمادا على خصائص معينة :

أحد المميزات الموجودة لدى JSON object أنها تتيح لك العمل بسهولة على خصائصها. سيتم استخدام عقدة التبديل (switch node) و التي تعمل كمفتاح أو توجيه للرسائل إلى اتجاه مخرج معين اعتمادا على خصائص الرسالة الواردة . على سبيل المثال، يمكنك فحص خصائص الحقل msg.payload.analyze و بالإعتماد على قيمتها (صحيحة / خاطئة ) ، تقرر توجيه الرسالة إلى أحد مخرجات عقدة التبديل (switch Node) .

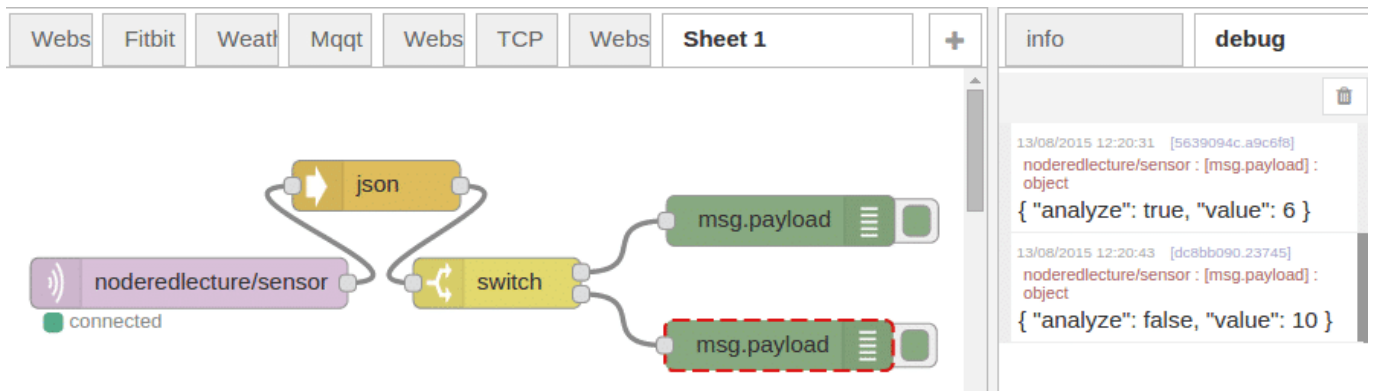
قم بسحب عقدة التبديل (switch node) ، ثم انقر مزدوجا عليها . قم بإعدادها للتحقق من قيمة الحقل msg.payload.analyze ، إذا كانت صحيحة سنقوم بإخراج الرسالة إلى المخرج الأول لعقدة التبديل ؛ و إذا كانت خاطئة سنقوم بإخراج الرسالة إلى المخرج الثاني لعقدة التبديل . قم بإعداد عقدة التبديل كما هو موضح بالصورة ادناه :

The 'Edit switch node' dialog box is shown. It has a 'Name' field with the placeholder 'Name'. Below it, the 'Property msg.' dropdown is set to 'payload.analyze'. There are two rules defined: the first rule is 'is true' which points to output '1', and the second rule is 'is false' which points to output '2'. There is a '+ rule' button to add more rules. At the bottom, there is a dropdown menu set to 'checking all rules'. At the very bottom of the dialog are 'Ok' and 'Cancel' buttons.

الآن يمكنك توصيل عقدتين إخراج debug كما هو موضح بالصورة التالية ، عند إعداد عدة مخارج للعقدة ، يتم ترقيم المخارج من أعلى ، لذلك المخرج 1 هو المخرج الأعلى و المخرج 2 هو المخرج السفلي في الشكل التالي.



قم بالرجوع إلى صفحة إدخال HiveMQ وإرسال رسالة {“analyze”:true, “value”:6} MQTT ، سوف تلاحظ أنه تم تفعيل المخرج الأول على عقدة التبديل و تم توجيه الرسالة الواردة على مخرج 1 ، إذا قمت بإرسال الرسالة التي تم استخدامها مسبقا {“analyze”:false, “value”:10} ، ستقوم عقدة التبديل (switch node) بتفعيل المخرج 2 و على عقدة الإخراج (debug) المربوطة على هذا المخرج ستظهر الرسالة. ستلاحظ على لوحة الإخراج عند تبويب debug على يمين النافذة طباعة الرسالة كما هو موضح بالصورة :

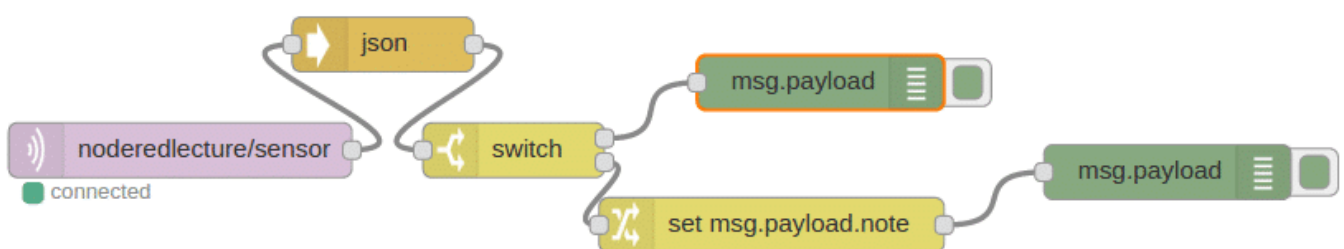


كما لاحظت، يوفر لك Node-RED عقد تسمح لك بسرعة تحديد محتويات الرسائل الواردة و توجيه الرسالة إلى أجزاء مختلفة من التدفق اعتمادا على محتوى معين للمدخلات.

## التلاعب في محتوى الرسائل بإستخدام عقدة التغيير :

عقدة أخرى مفيدة وهي عقدة التغيير (change node) ، و التي تسمح لك لتغيير محتوى الرسالة أو إضافة خصائص جديدة عليها . فيمكنك استخدام هذه العقدة للتأثير على الخصائص في الرسالة إما عن طريق تغيير خصائص موجودة ، أو حذف أو إضافة خصائص جديدة.

في هذا المثال ، سيتم إضافة حقل جديدة للرسالة وهو msg.payload.note . أولاً، قم بسحب و إسقاط عقدة التغيير (change node) و توصيلها إلى المخرج الثاني من عقدة التبديل كما في الشكل أدناه ، كما نذكرون، ان محتوى حقل msg.payload.analyze الرسالة هو false فيتم اخراجها على المخرج رقم 2.



الآن قم بإعداد العقدة لتعيين msg.payload.note لتحتوي على "this is not being analyzed" كما مبين لك بالصورة التالية :

### Edit change node

**Name**

**Rules**

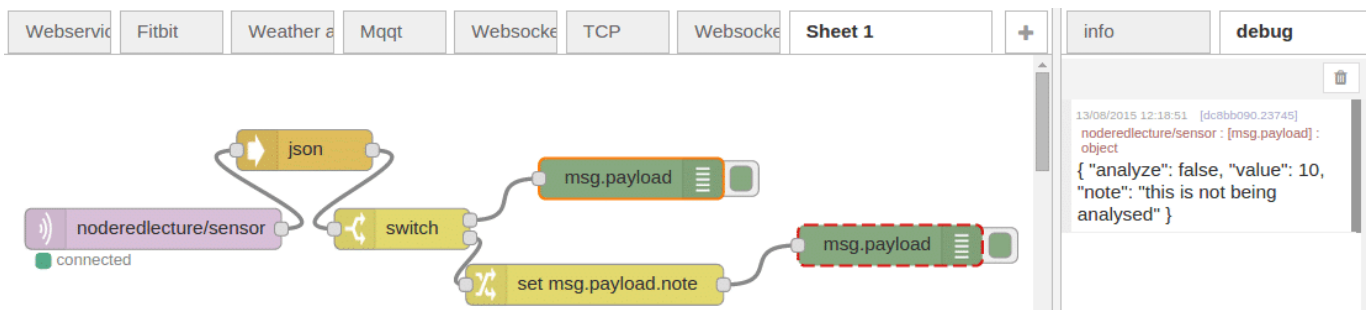
Set
msg. payload.note
✕

to
this is not being analyzed

+ rule

Ok Cancel

عند استقبال الرسالة التي ترسلها عقدة التبديل (switch node) إلى المخرج 2 ، سوف يتم تعديلها لتحتوي على عنصر "note" مع جملة "this is not being analyzed" . اذا قمت بنشر (deploy) واختبار التدفق بواسطة ارسال رسالة MQTT من HiveMQ ، سوف يظهر لك المخرج كما هو موضح بالصورة التالية:

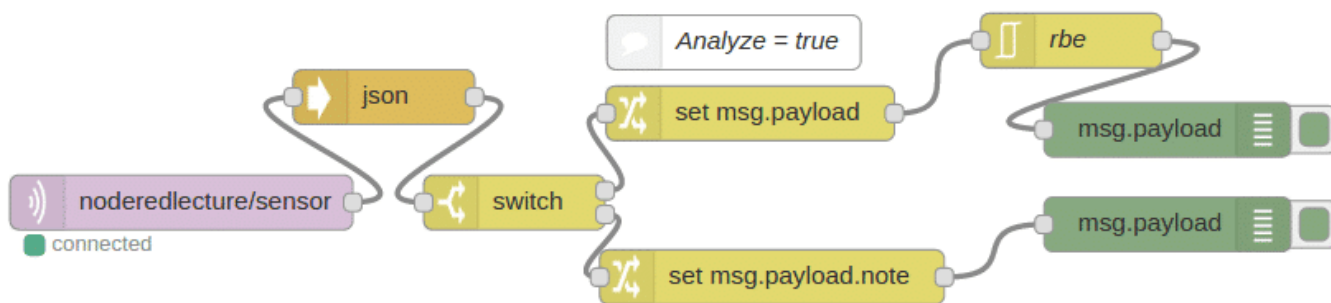


## استخدام عقدة RBE :

في هذا المثال، سيتم إضافة عقدة (report by exception -RBE) والتي تقوم بتمرير البيانات فقط اذا طرأ عليها تغيير . يمكنك تعيينها لفحص حمولة الرسالة (message payload) وإما عن طريق تجاهلها حتى يطرأ أي تغيير على الرسالة (في حالة RBE) أو عندما تتغير الرسالة بنسبة معينة (في حالة deadband) .

في حالة استخدام RBE ، تعمل على الأرقام و الجمل . وفي حالة استخدام deadband فهي تعمل على أرقام فقط و يستخدم في إعداد الـ deadband نطاق + أو - ، بحث تكون القيمة الواردة يمكن أن تتقلب داخل نطاق معين قبل أن تنطلق .

سنقوم بإضافة عقدة تغيير (change node) والتي سيتم توصيلها على مخرج 1 لعقدة التبديل (switch node) . ثم سنقوم بتوصيل عقدة rbe إلى عقدة التغيير كما هو موضح بالصورة أدناه . تذكر بأن الناتج يتوافق مع "analyze" ، وقد تم إضافة عقدة تعليق (comment node) و كتب عليها "analyze=true" ، لأنها مفيدة عند عمل تدفقات معقدة .



قم بفتح محرر عقدة التغيير (change node) لتعيين msg.payload إلى msg.payload.value . وهكذا يحدد المخرج من هذه العقدة إلى القيمة الموجودة في العنصر msg.payload.value من المدخلات الواردة كما في الشكل أدناه .

### Edit change node

**Name**

**Rules**

Set	msg. payload	x
	to	msg.payload.value

+ rule

Ok

Cancel

سنقوم بتوجيه الرسالة إذا تم تغيير القيمة بنسبة 20% أو أكثر من ذلك ، فسوف تحتاج إلى النقر نقرًا مزدوجًا فوق عقدة RBE و لتجاهل الرسالة ما لم تتغير القيمة بأكثر من 20 % .

### Edit rbe node

**Mode**

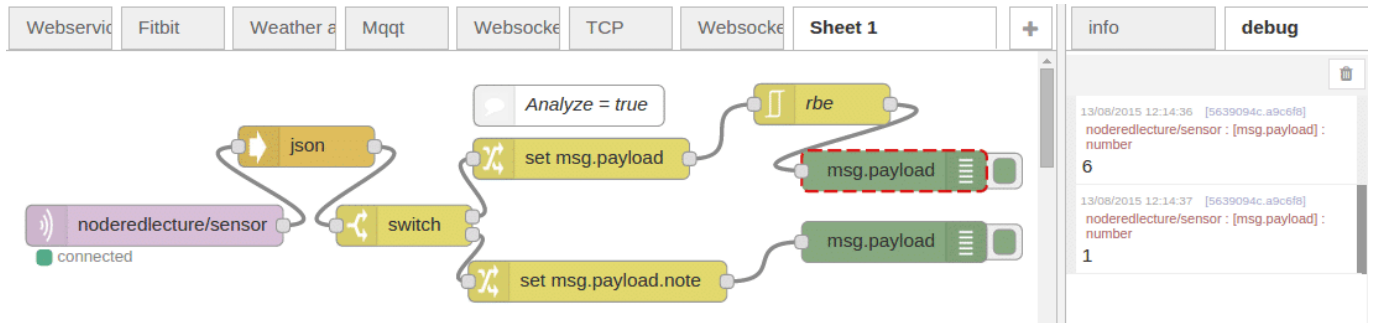
**Name**

Ok

Cancel

لإختبار التدفق، قم بنشر deploy هذا التدفق ومن ثم العودة إلى صفحة HiveMQ و إرسال سلسلة من الرسائل. أولاً ، سوف تحتاج إلى تعيين قيمة analyze إلى true و بالتالي تقوم عقدة التبديل بإرسال الرسالة عبر المخرج الأول. إذا كنت تستخدم الرسالة التي تم تعيينها

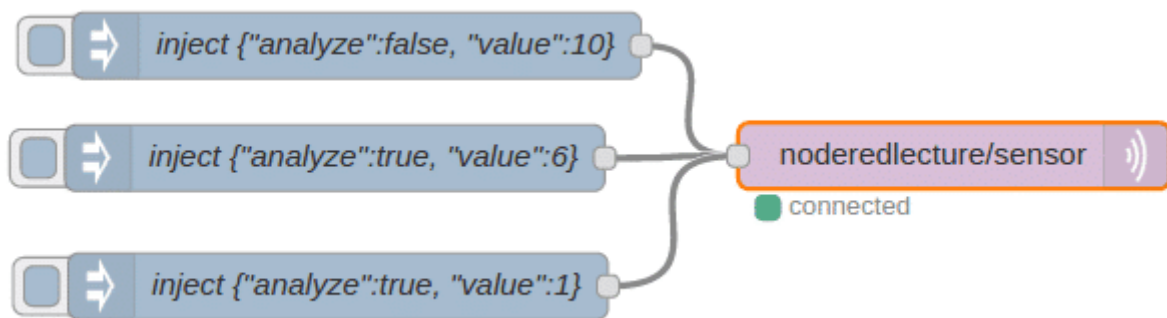
سابقة فإن القيمة (value) من 6 ، ستفشل في عبور عقدة RBE . ثم قم بإرسال رسالة ثانية من قيمة 10 ، فإن العقدة RBE تقييم الفرق ما بين 6 و 10 ، ويبرها اكبر من 20 % ، فيقوم بإرسال رسالة إلى عقدة debug النهائية والتي ستطبع على لوحة الإخراج debug كما هو موضح بالصورة التالية :



## استخدام عقدة الإخراج MQTT لإختبار التدفق :

كبدل لاستخدام صفحة الاختبار HiveMQ لنشر موضوع MQTT ، سنقوم بإعداد و تكوين عقدة اخراج mqtt . وهي تسمح لك لإعداد/تكوين خدمة MQTT و الموضوع الذي تريد نشره عليه. ويمكنك بعد ذلك إرسال عقدة الرسائل كسلسلة JSON كما كانت ترسل عبر صفحة الاختبار HiveMQ .

لعمل على ذلك، قم بسحب و إسقاط ثلاث عقد inject و عقدة اخراج MQTT كما هو موضح بالشكل التالي:



يمكنك الآن اختبار التدفق الذي قمت بإنشائه لتحليل رسالة MQTT مباشرة من مساحة العمل عن طريق النقر على زر الضحك على الجانب الأيسر لعقد inject الثلاثة.

## websockets

مآخذ الويب (Websockets) هي وسيلة أخرى مفيدة للقدرة على التواصل وهي مبينة ضمن عقد Node-RED عبر عقدة websocket . توفر مآخذ الويب (websocket) اتصال duplex TCP وقد صممت للسماح لمتصفحات الويب و الخوادم للحفاظ على "backchannel" والتي يمكن استخدامها لزيادة تفاعلات HTTP التقليدية ، مما يسمح للخوادم لتحديث صفحات الويب دون طلب العميل .

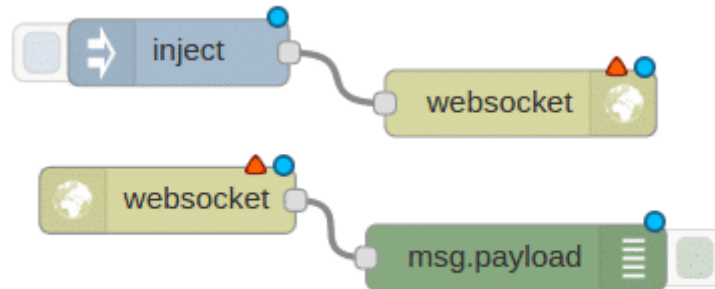
وعقدة websocket اما تكون للإدخال أو الإخراج ، مما يتيح لك الاستماع للبيانات الواردة (المدخلات) أو إرسال (الإخراج) على websocket . تم تصميم عقدة الإخراج للتحقق وللمعرفة ما اذا نشأت حمولة الانتاج على websocket في العقدة ، وفي هذه الحالة فإنه يستجيب إلى المرسل الأصلي . وإلا فإنه سوف يقوم ببث الحمولة لجميع مآخذ الويب .

وبالإضافة إلى ذلك، يمكن إعداد كل من المدخلات و المخرجات لعقدة websocket إما أن تكون خادم أو عميل - في وضع الخادم

سيقوم بالإستماع إلى عنوان URL ، وفي وضع العميل سوف يتصل إلى عنوان IP معين .

لمعرفة كيفية عمل عقدة websocket ، عليك استخدام مآخذ الويب العامة التي تعمل على الموقع التالي:  
(<https://www.websocket.org/echo.html>).

قم بسحب عقدة inject و عقدة إدخال و عقدة إخراج websocket و عقدة التصحيح debug ، ثم قم بتوصيل العقد كما هو موضح بالصورة التالية :



قم بإعداد عقدة inject لإرسال جملة payload تحتوي على "Hello There" كما في الصورة التالية :

**Edit inject node**

Payload

string

Topic

Hello There

Repeat

none

☐

Inject once at start?

Name

Name

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

Ok

Cancel

قم بإعداد عقد websocket للإتصال إلى إرسال `wss://echo.websocket.org` كما هو موضح بالصورة :



## Edit websocket out node

Type Connect to

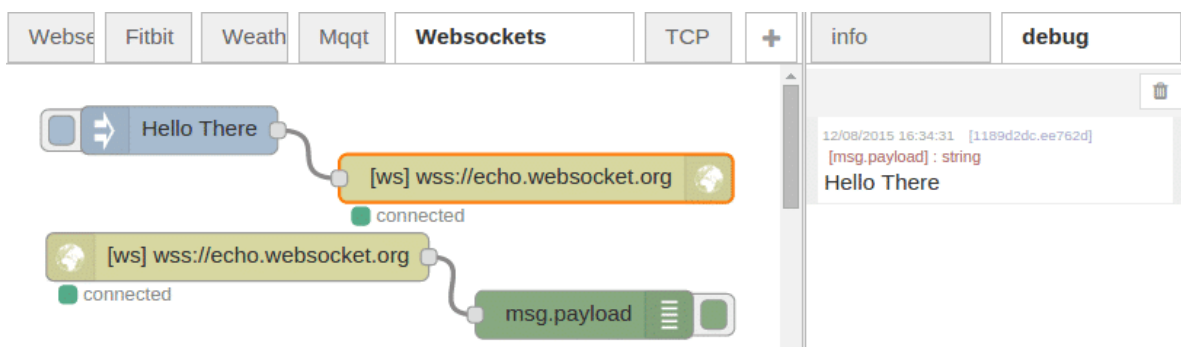
URL wss://echo.websocket.org

Name Name

Ok

Cancel

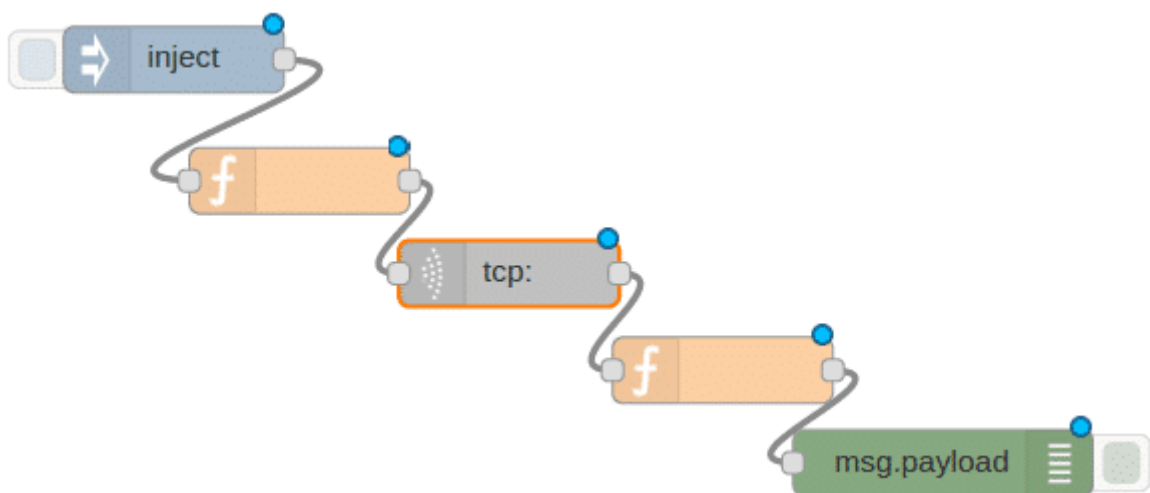
قم بنشر deploy التدفق ، عندما يتم الضغط على زر ضح عقدة inject سترى طباعتها كما هو موضح بالصورة :



## إرسال طلبات TCP :

يوضح لك هذا المثال كيفية إرسال طلبات TCP باستخدام عقدة tcp . في هذه الحالة سوف تقوم بعمل طلب HTTP إتباعا للمواصفات في (<http://tools.ietf.org/html/rfc2616#section-5.1.2>).


يوضح هذا المثال استخدام عقدة tcp . حيث يمكن تكوين عقدة UDP أو HTTP لطريقة مماثلة . للبدء، دعونا نقوم بتوصيل عقد ال inject و function و tcp و debug كما هو موضح بالصورة أدناه :




قم بتعديل على عقدة function للإضافة دالة و التي تضح جملة "GET / HTTP/1.1\r\n\r\nHost: www.google.com" كحمولة (payload) كما هو موضح بالصورة أدناه.


هذه سلسلة الجملة عبارة عن طلب HTTP ، مشيراً إلى أنه طلب GET ، و البروتوكول هو HTTP 1.1 و المضيف (host) هو www.google.com . و `\r\n\r\n` هو زوج من إرجاع/السطر الجديد (return/newline) و الذي يكون مطلوب في بروتوكول HTTP .

**Edit function node**


 Name

Set GET request string



 Function

```
1 msg.payload = "GET / HTTP/1.1\r\n\r\nHost: www.google.com"
2 return msg;
```

 Outputs

1


See the Info tab for help writing functions.

Ok

Cancel

قم بإعداد عقدة طلب tcp للاتصال بالخادم www.google.com ، على المنفذ 80. و الإعداد لإغلاق الاتصال بعد 1 ثانية (1000 ميلي ثانية ) كما هو موضح بالصورة التالية :


**Edit tcp request node**

 Server

www.google.com

port


80

 Return


after a fixed timeout of

1000

ms

 Name

Name



Ok

Cancel

إذا قمت بنشر deploy التدفق و الضغط على زر الضح في عقدة inject ، فسوف تقوم بعمل طلب إلى google و سوف تحصل على استجابة TCP . عقدة التصحيح debug ستقوم بطباعة الإستجابة كجملة كما هو موضح بالصورة التالية :

## Edit function node

Name

Parse response buffer into string

Function

```
1 msg.payload = msg.payload.toString('utf8');  
2 return msg;
```

Outputs

1

See the Info tab for help writing functions.

Ok

Cancel

إذا قمت بنشر deploy التدفق و الضغط على زر الضحك في عقدة inject ، فسوف تقوم بعمل طلب إلى google و سوف تحصل على استجابة TCP . عقدة التصحيح debug ستقوم بطباعة الإستجابة كجملة كما هو موضح بالصورة التالية :

The screenshot shows a Node-RED flow with the following nodes:

- inject**: A blue node with a button icon.
- Set GET request string**: An orange function node with the label "f".
- tcp:www.google.com:80**: A grey node with a network icon.
- Parse response buffer into string**: An orange function node with the label "f".
- msg.payload**: A green node with a list icon.

The debug console on the right shows the following output:

```
12/08/2015 16:46:35 [9e76964b.718968]  
[msg.payload] : string  
HTTP/1.1 200 OK Date: Wed, 12 Aug 2015 23:46:34 GMT Expires: -1  
Cache-Control: private, max-age=0 Content-Type: text/html;  
charset=ISO-8859-1 P3P: CP="This is not a P3P policy! See  
http://www.google.com/support/accounts/bin/answer.py?  
hl=en&answer=151657 for more info." Server: gws X-XSS-Protection: 1;  
mode=block X-Frame-Options: SAMEORIGIN Set-Cookie:  
PREF=ID=1111111111111111:FF=0:TM=1439423194:LM=1439423194:V=  
expires=Fri, 11-Aug-2017 23:46:34 GMT; path=/; domain=.google.com  
Set-Cookie:  
NID=70=MDGgTJ19yvd7rG3XdR4sTijHP0lNEY01zdGyMnJCKkbes31UrE  
expires=Thu, 11-Feb-2016 23:46:34 GMT; path=/; domain=.google.com;  
HttpOnly Accept-Ranges: none Vary: Accept-Encoding Transfer-  
Encoding: chunked 8000 <!doctype html><html itemscope=""  
itemtype="http://schema.org/WebPage" lang="en"><head><meta  
content="Search the world's information, including webpages, images,  
videos and more. Goo ....
```