مشاريع Node-RED متوسطة التدفق (الجزء الثاني)

المقدمة

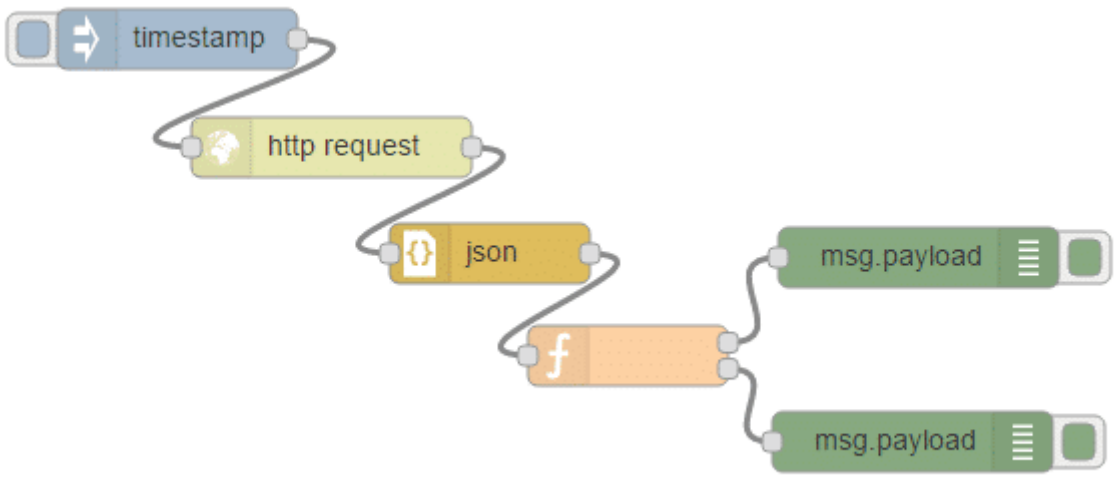
في هذا الدرس، سوف نبني على الأفكار التي تم عرضها في المحاضرة السابقة و التركيز على الأمثلة التي تكشف بعض المفاهيم الأساسية من الدرس السابق. و يشمل هذا الدرس مزيد من الأفكار مثل ارسال رسائل متعددة على مخرج واحد أو انشاء موقع مدونة خاص بك.

الحصول على بيانات الزلازل من API الخارجية و إعادته كرسائل متعددة:

يوضح هذا المثال كيفية الحصول على بيانات من API الخارجية وكيفية فصل تلك البيانات باستخدام عقدة function. سوف نستخدم البيانات من API الخارجية والتي توفر الوصول إلى بيانات الزلزال التي يتم توفيرها من قبل المسح الجيولوجي الأمريكي (USGS) .

(http://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php)

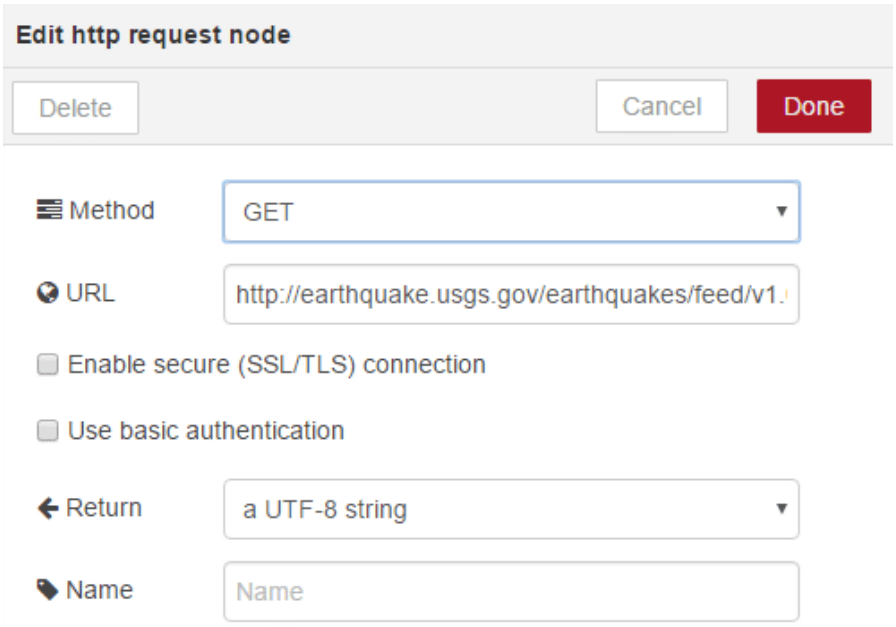
أولا، دعونا نقوم بإنشاء التدفق كما هو بالشكل التالي:



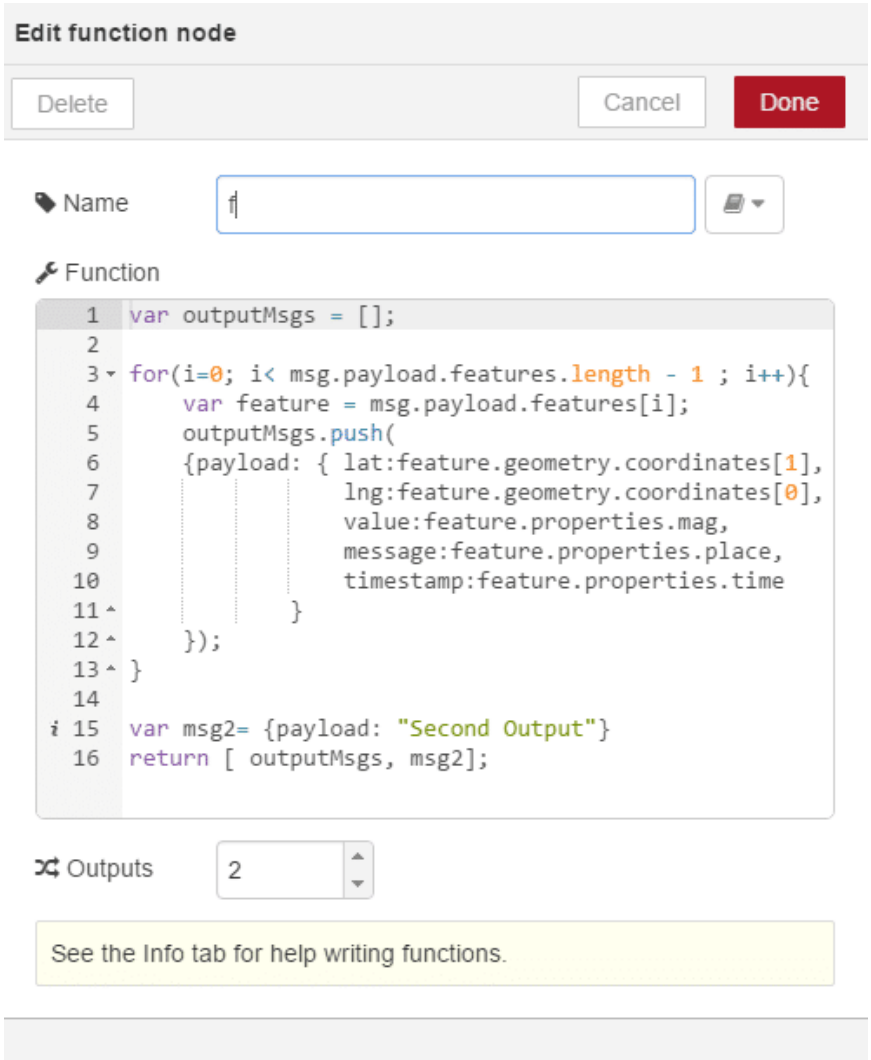
قم بالتعديل على عقدة http request للحصول على البيانات من عنوان URL التالي كما هو موضح بالصورة أدناه:

http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_month.geojson

تحتوي هذه البيانات على الزلازل من الشهر الماضي. بعد طلب البيانات، عقدة JSON سوف تسمح لك بتحليل محتوى الاستجابة إلى كائن قابل للاستخدام بواسطة عقدة Function التالية .



الآن قم بالتعديل على عقدة function وإضافة التعليمات البرمجية كما هو مبين أدناه :



شرح الكود :

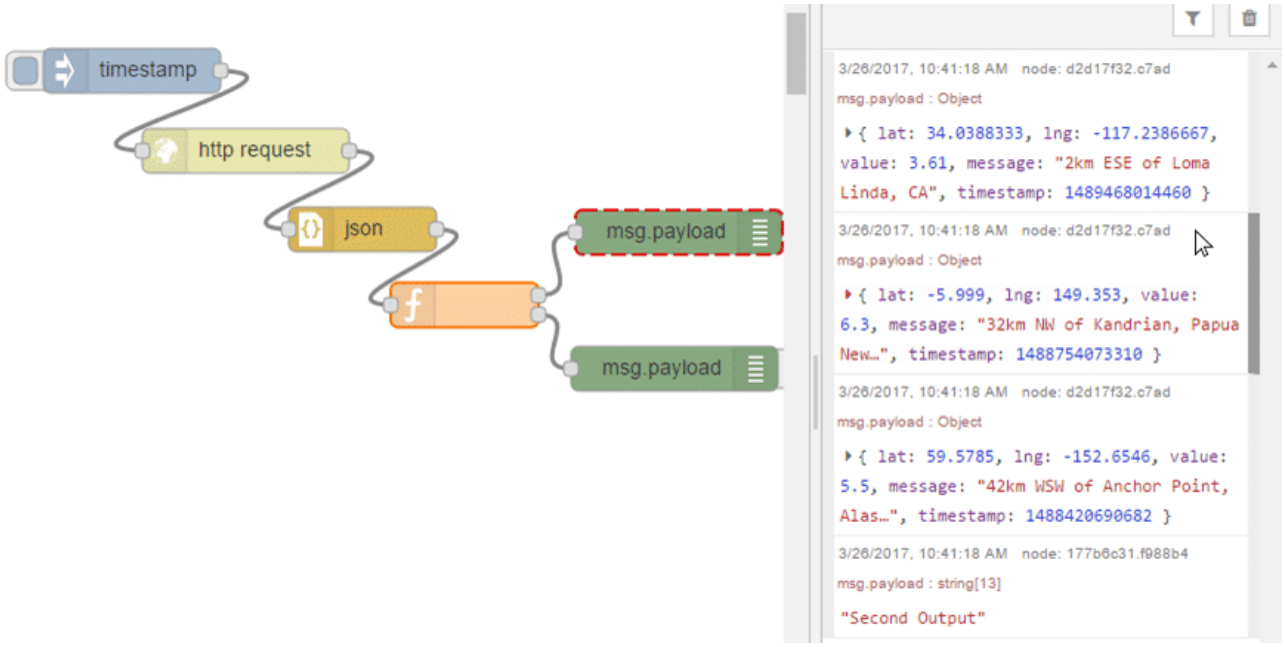
سيؤدي هذا الكود إلى إنشاء مصفوفة (outputMsgs) تحتوي على مصفوفة من الرسائل التي تم إنشاؤها من بياناتك. تمر حلقة التكرار (For loop) من خلال استجابة JSON وتقوم بإنشاء رسالة جديدة تحتوي على حمولة تحتوي على خط العرض (lat) ، وخط الطول

(lng) ، القيمة (value) ، الطابع الزمني (timestamp). ثم سيتم تدفع كائن الرسالة الجديدة إلى مصفوفة outputMsgs .

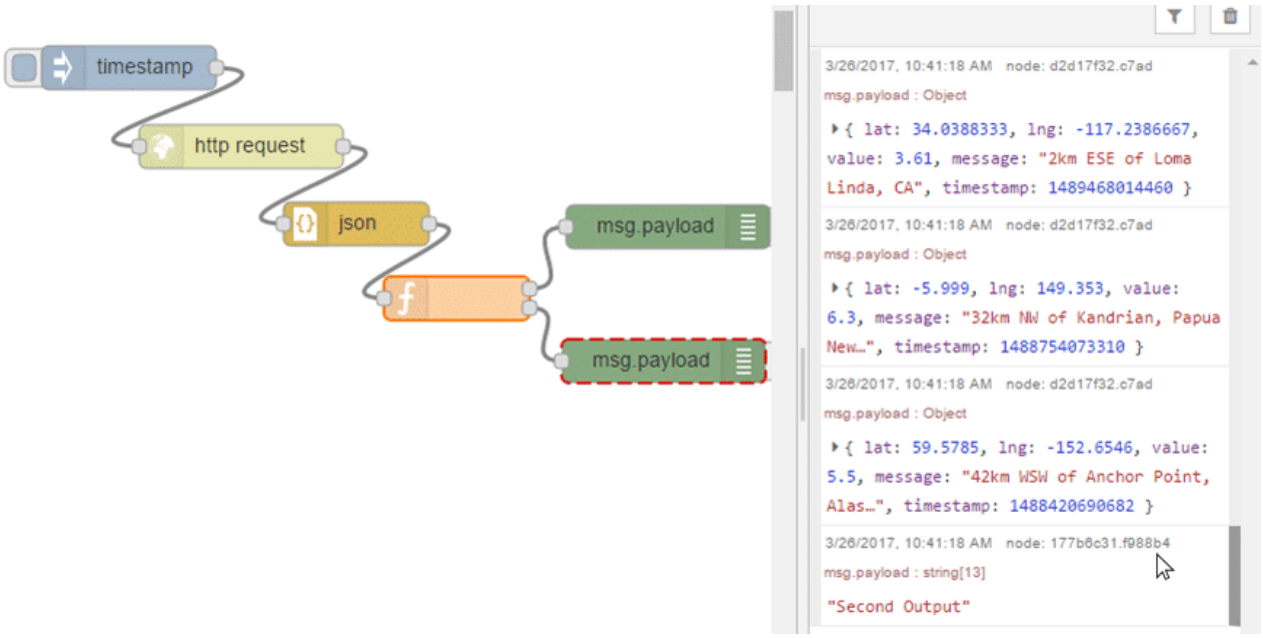
فإنه سيتم إنشاء متغير جديد يدعى (msg2) مع كائن رسالة جديدة تحتوي على عنصر الحمولة مع سلسلة "Second Output".

وأخيرا، ستقوم الدالة (function) بإرجاع مصفوفة تحتوي على عنصرين. العنصر الأول هو مصفوفة من الرسائل (outputMsgs); و العنصر الثاني هو رسالة واحدة (msg2). لذلك سنقوم بإعداد عقدة function ليكون لديها مخرجين . لتتناسب مع مجموعة العناصر التي يتم إرجاعها.

عند النقر فوق الزر الموجود على عقدة inject، يمكنك أن ترى الإخراج العلوي يحتوي على مصفوفة من الرسائل التي تم تحليلها من البيانات. هذا يتوافق مع مصفوفة outputMsgs .



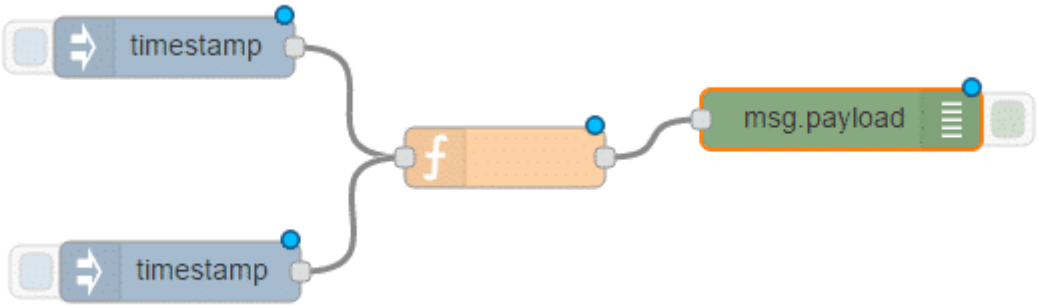
والمخرج الثاني يحتوي على رسالة واحدة (msg2) :



مدخلات متعددة على عقدة function :

تم تصميم عقد function في Node-RED لمعالجة الرسائل ككيانات واحدة. ومع ذلك، قد تعتمك وظائفك في بعض الحالات على مصدرين منفصلين للبيانات. هناك العديد من الطرق للتعامل مع هذه الحالات في Node-RED . يستخدم النهج التالي كائن السياق (context) في Node-RED و الموضوعات للسماح لدالة الانتظار (function wait) لعدة رسائل للوصول من أجل العودة. لقد تم عرض كيفية إعداد و استخدام بيانات السياق في الدروس السابقة .

سنبدأ من خلال ربط عقدتين inject، و عقدة Function ، وعقدة تصحيح كما هو موضح بالشكل التالي :



سنقوم بتعديل عقدة Function وإضافة التعليمات البرمجية كما هو موضح أدناه. سيستخدم هذا الكود كائن السياق في Node-RED وإضافة عنصر البيانات.

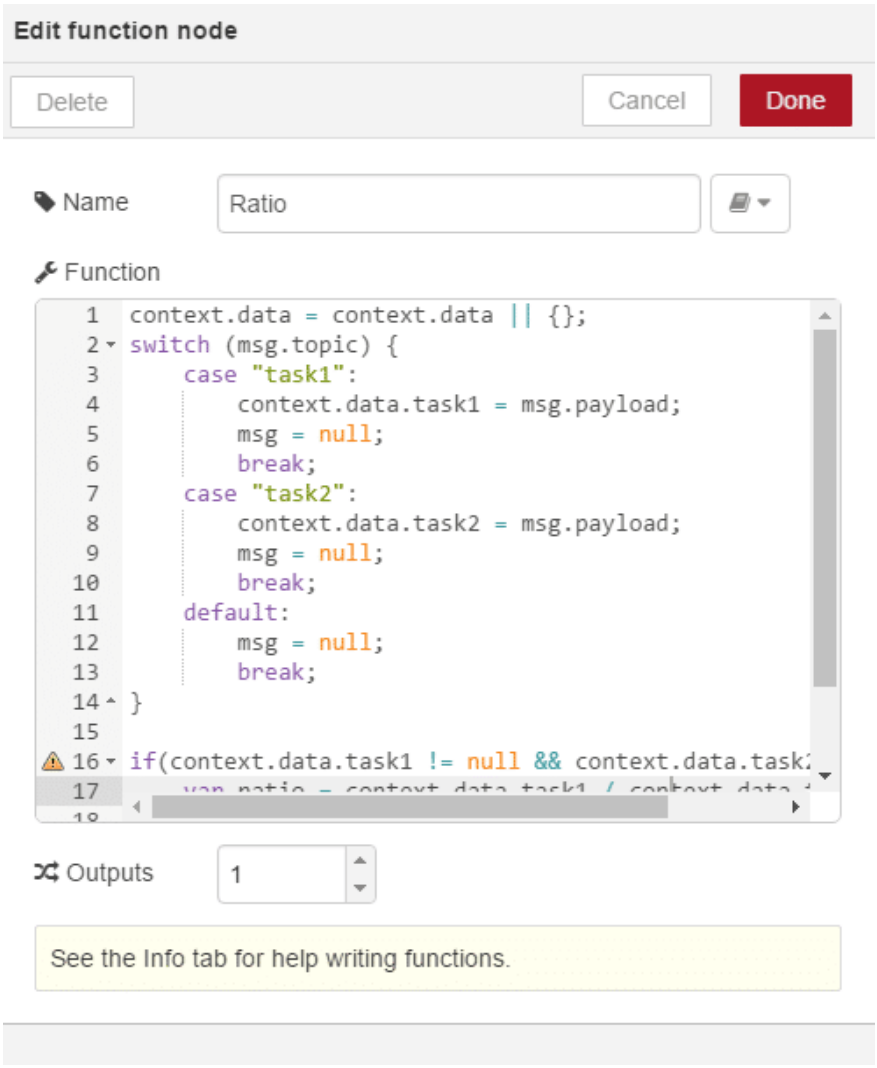
```
context.data = context.data || {};
switch (msg.topic) {
    case "task1":
        context.data.task1 = msg.payload;
        msg = null;
        break;
    case "task2":
        context.data.task2 = msg.payload;
        msg = null;
        break;
    default:
        msg = null;
        break;
}

if(context.data.task1 != null && context.data.task2 != null){
    var ratio = context.data.task1 / context.data.task2;
    context.data=null;
    return {payload:ratio};
}else return msg;
```

شرح الكود:

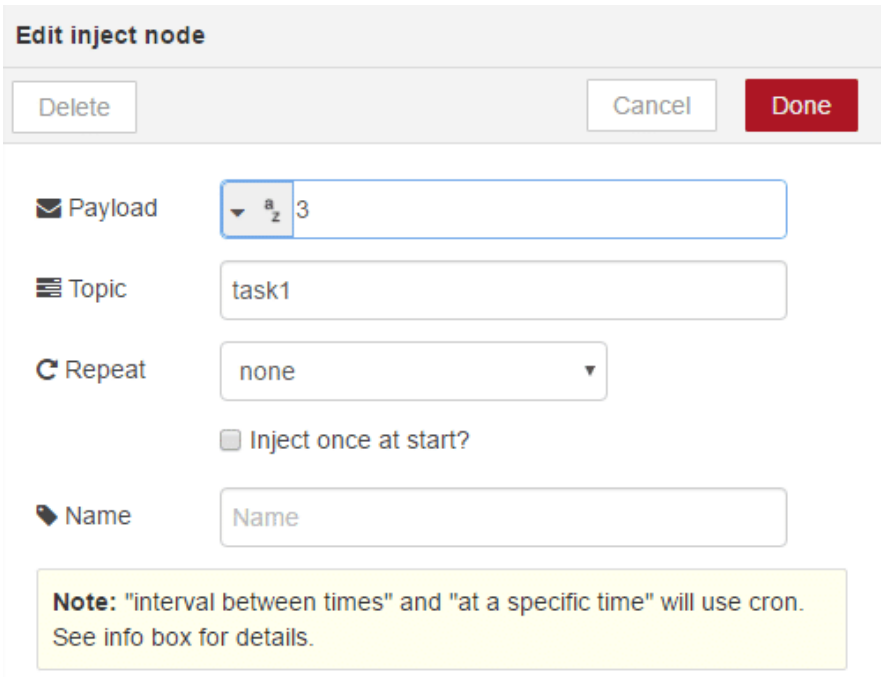
السطر الأول، لتهيئة كائن السياق. ثم يتم استخدام switch statement في السطر 2 ليتم البحث عن حقل الموضوع في الرسالة. يستخدم هذا لتعيين حقل task1 أو task2 للكائن context.data . وبالتالي يتم تجاهل أي موضوع رسالة أخرى. ثم يتحقق الشطر 16 لمعرفة ما إذا كانت الدالة قد تلقت رسائل من كلا النوعين من المواضيع (task1 and task2). إذا لم يكن الأمر كذلك، ترجع الدالة

رسالة خالية (null) وتعود إلى انتظار رسالة أخرى. وإلا فإن السطر 17 يحسب النسبة وينتجها كرسالة.

قم بكتابة الكود اعلاه على عقدة Function كما هو موضح بالشكل التالي :

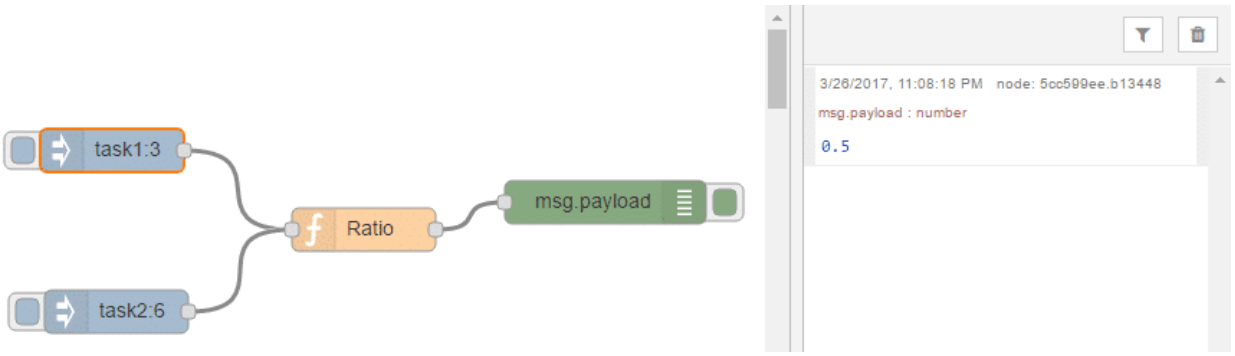


قم بإعداد عقدة inject الأولى لإرجاع حمولة تحتوي على سلسلة من "3" ، مع موضوع "task1" .



يجب عليك ايضا إعداد عقدة inject الثانية لإرجاع سلسلة الحمولة من "6" ، مع موضوع"task2" .

يمكنك بعد ذلك نشر التدفق (deploy) . قم بالنقر على الزر الأيسر لعقدة inject الأولى (task1: 3) . ستلاحظ ظهور رسالة تشير أن السلسلة تم ضخها عبر التدفق بنجاح، ولكن لن ترى أي شي على لوحة الإخراج عند تبويب debug . ثم قم بالنقر على الزر الأيسر لعقدة inject الثانية (task2: 6) . ستظهر رسالة تشير بنجاح ضخ الرسالة، كما سيظهر على لوحة الإخراج النسبة كما هو موضح بالشكل التالي :

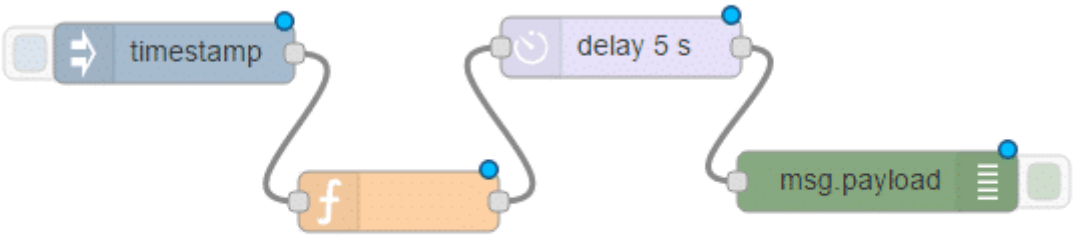


السماح لعقدة function بإرسال رسائل متعددة على مخرج واحد :

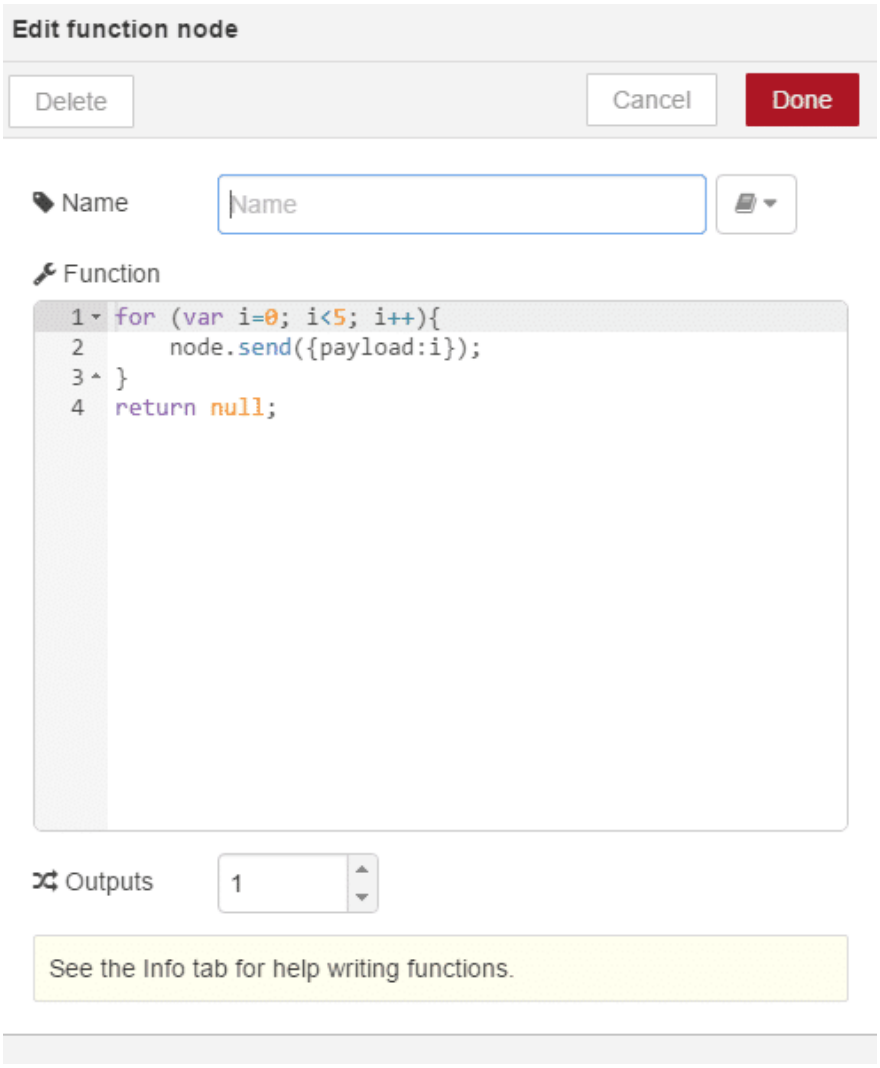
في الدرس يوضح كيفية إعداد وإرسال الرسائل على عقد إخراج متعددة. يوضح هذا المثال كيفية إرسال رسالة متعددة، ولكن على نفس المخرج، من عقدة function واحدة.

ويستند المثال التالي على تدفق من قبل Node-RED contributor dceejay . يمكنك الحصول على التدفق الأصلي في http://flows.nodered.org/flow/5c4c0f5a08d4e91ea14d

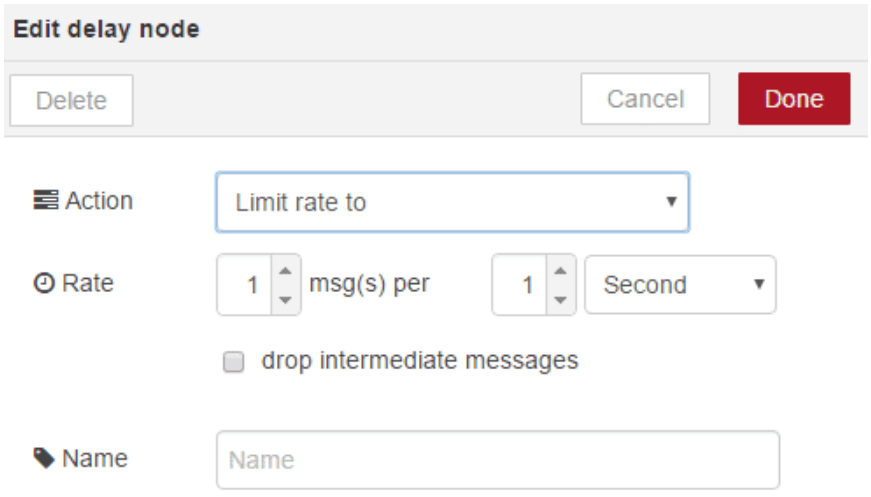
أولا، قم بربط عقدة inject و function و delay و debug كما هو موضح :



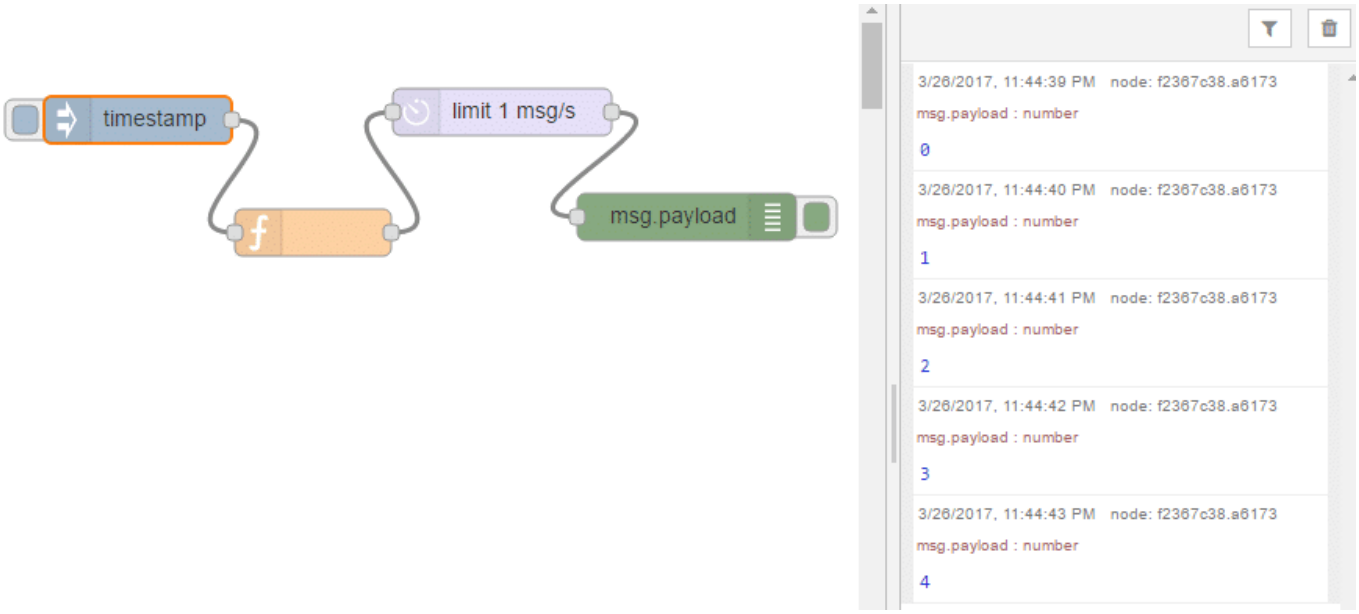
قم بالتعديل على عقدة function و إضافة الكود كما هو مبين بالصورة أدناه. هذا الكود هو بسيط جدا. فهو يحتوي على حلقة التكرار التي تستدعي "node.send" خمس مرات . تم استخدام node.send في الدرس . فهي تسمح لعقدة function لإخراج الرسائل المنتجة على مخارجها بشكل مستقل عن قيمة الإرجاع. لاحظ فإن الدالة تقوم بترجيع Null . والانتاج المتوقع لهذه الدالة هي سلسلة من الرسائل مع الحمولات :0،1،2،3 و 4 .



ولجعل المثال يعيد تجميع عداد أكثر دقة، يمكنك تعديل عقدة التأخير و تهيئته لتقييد معدل الرسائل(limit the rate) إلى 1 في الثانية، كما هو مبين بالشكل التالي:



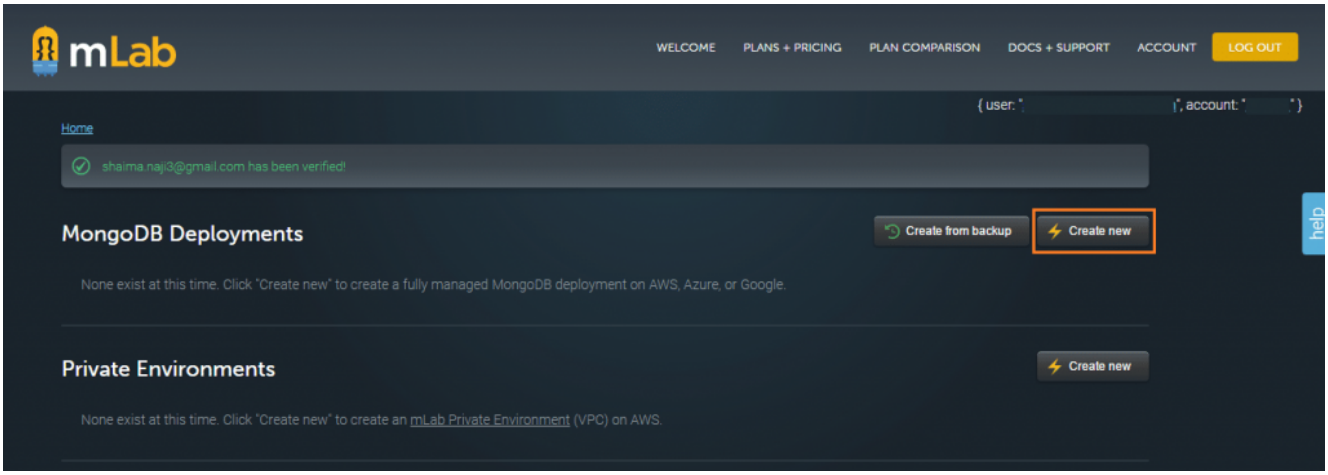
قم الآن بنشر التدفق، ثم الضغط على الزر الأيسر لعقدة inject وفحص لوجة الإخراج عند تبويب debug ، ستلاحظ وصول الرسائل كل ثانية، انظر الشكل التالي :



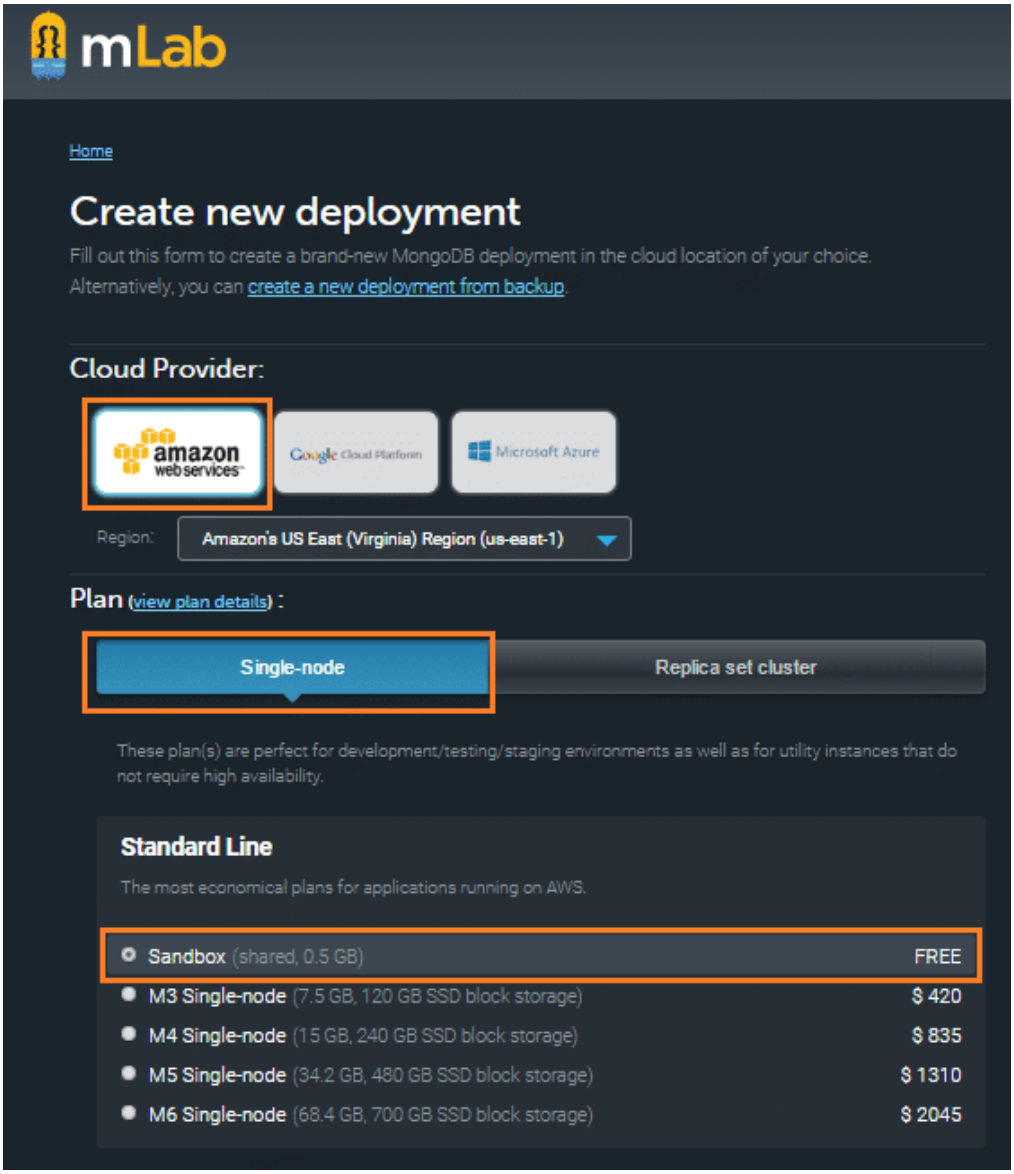
إنشاء موقع مدونة باستخدام Node-RED :

من خلال هذا المثال، سوف يظهر لك كيفية بناء خدمة المدونات الصغيرة مع عدد قليل من العقد في Node-RED . سنقوم بإستخدام عقد MongoDB كتخزين للمشاركات، وعقد http لتوفير نقاط النهاية للخدمة وعقدة html لتنسيق صفحة ويب للمدونة.

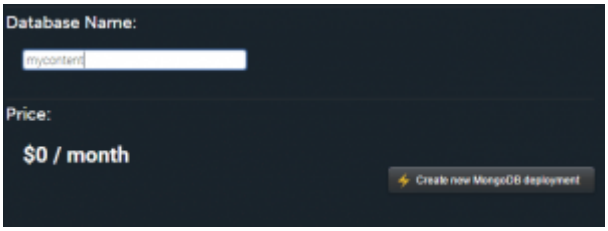
سنقوم بإستخدام خدمة السحابة المجانية التي تدعى "mongolab" . حيث انها تسمح لك لإنشاء قواعد البيانات "sandbox" التي يمكن استخدامها لنموذج التطبيقات الخاصة بك. قم بفتح الموقع https://mlab.com، ثم قم بالتسجيل للحصول على حساب. ستتمكن بعد ذلك من إنشاء قاعدة البيانات. قم بالنقر على إنشاء جديد "Create New" في لوحة التحكم الرئيسية.



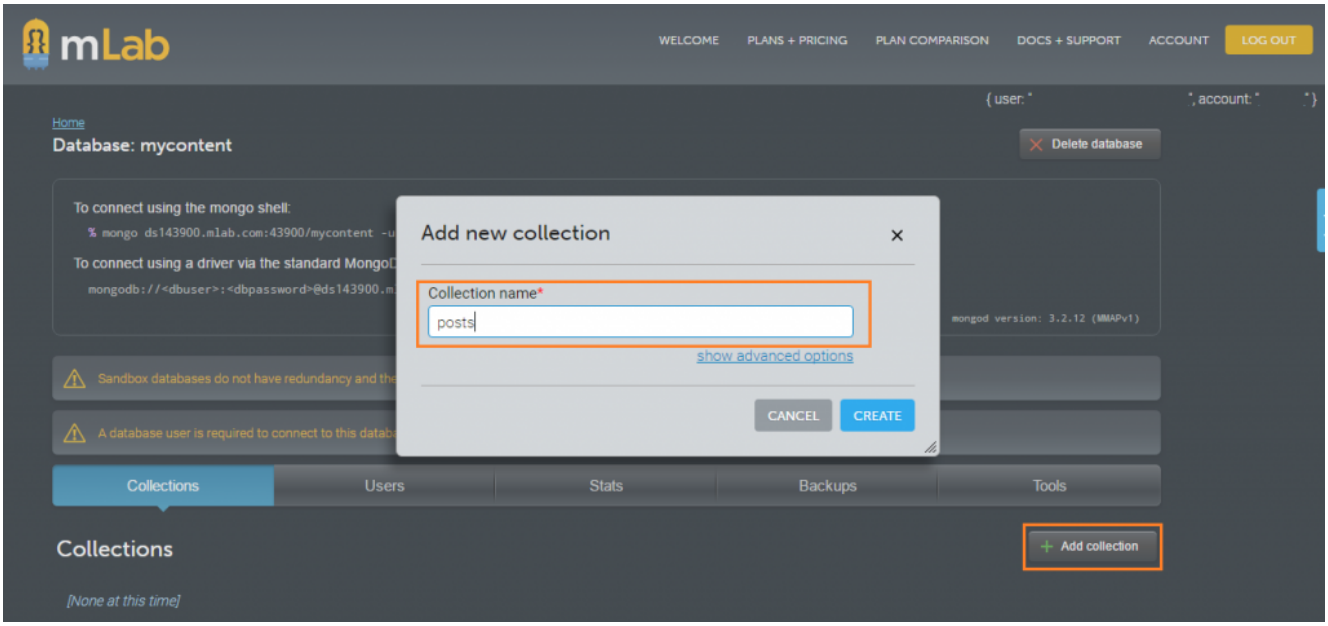
قم بملأ النموذج لإنشاء نشر جديد من MongoDB . قم بتحديد أمازون (Amazon)، ثم قم بإختيار Single-node وحدد الحجم المجاني وهو 500MB .



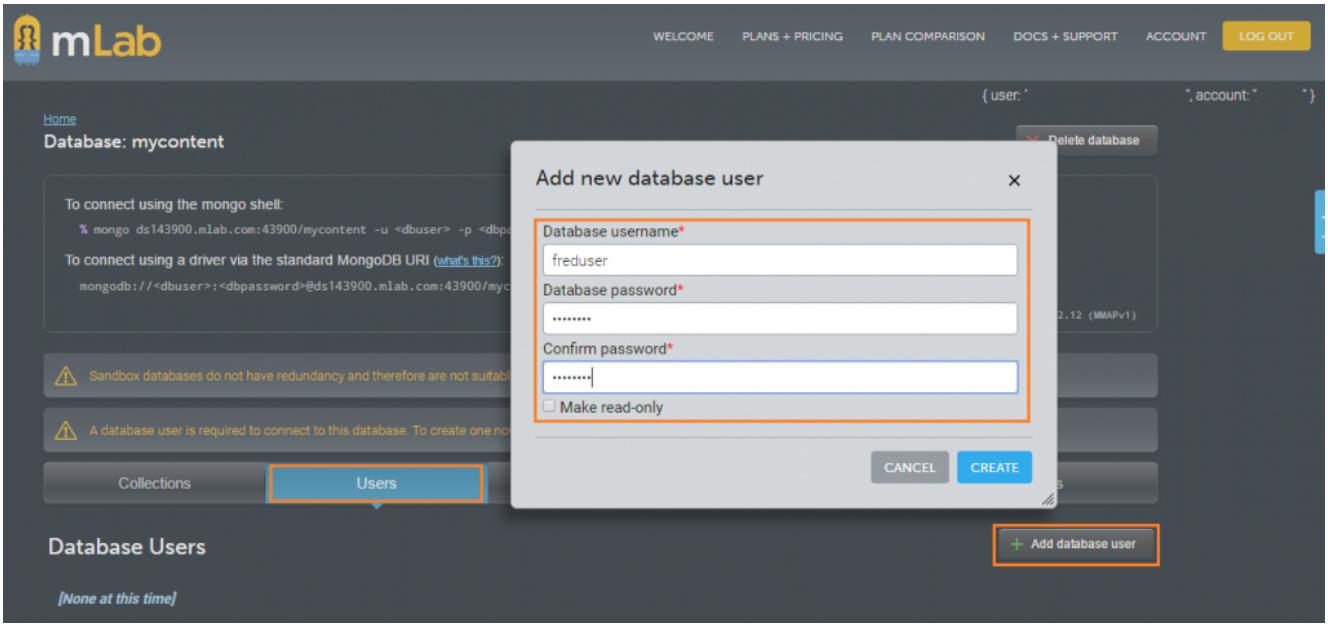
ثم قم بتسمية قاعدة البيانات. سنقوم بتسميتها في هذا المثال "mycontent" . الآن قم بالنقر على إنشاء نشر MongoDB جديد :



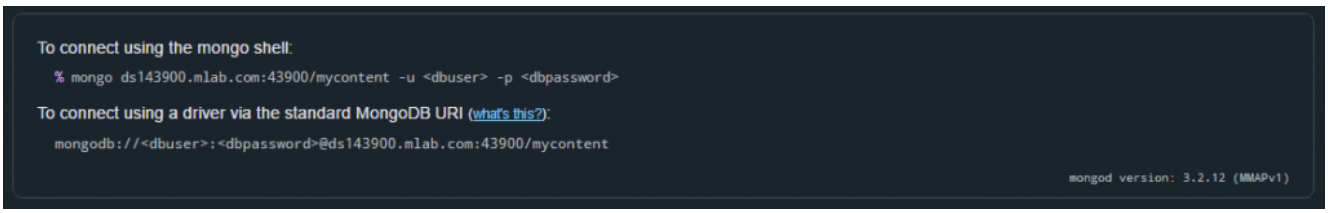
يسمح لك MongoDB لإنشاء مجموعات "collections" لكل قاعدة بيانات. وهي مماثلة للجداول في قواعد البيانات العلائقية، كل مجموعة "collection" تحمل الوثائق "documents" التناظرية إلى السجلات "records" في قواعد البيانات العلائقية. من أجل البدء في استخدام قاعدة البيانات الخاصة بك تحتاج إلى إنشاء مجموعة جديدة. حدد قاعدة البيانات التي تم إنشاؤها حديثا في لوحة التحكم وأضف مجموعة جديدة وسوف نقوم بتسميتها بـ "posts" .



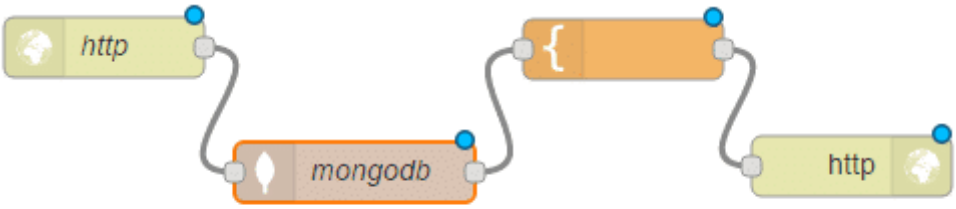
وأخيرا، سوف تحتاج مستخدم للإتصال بقاعدة البيانات تلك. قم بإضافة مستخدم قاعدة بيانات جديدة. وسوف نقوم بتسميتها "freduser" .



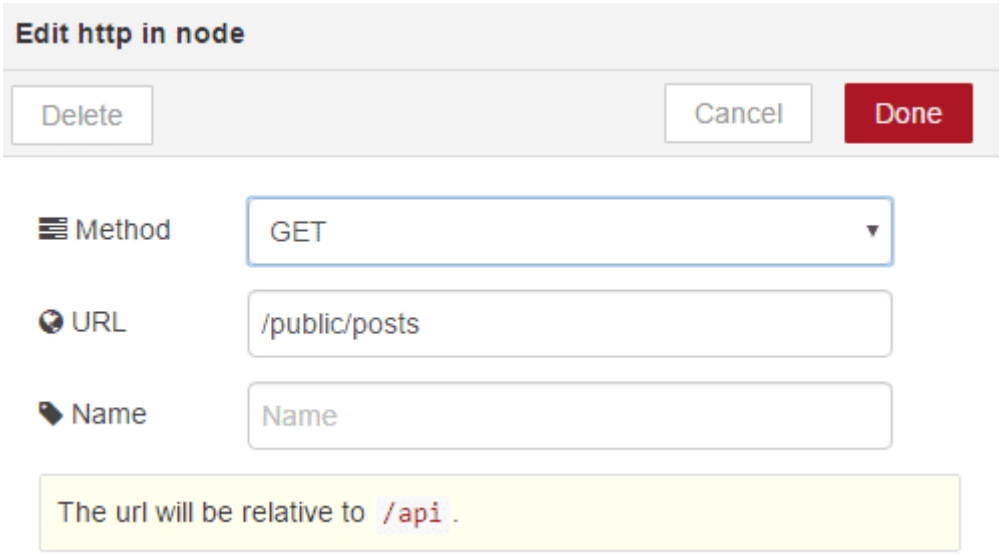
وأخيرا، لاحظ كيف ستوفر لك هذه الصفحة معلومات هامة حول كيفية الاتصال بقاعدة البيانات الخاصة بك. سنحتاج إلى عنوان URI (في حالتنا ds143900.mlab.com)، والمنفذ (في حالتنا 43900) والمستخدم وكلمة المرور الذي تم إنشاؤهما حديثا.



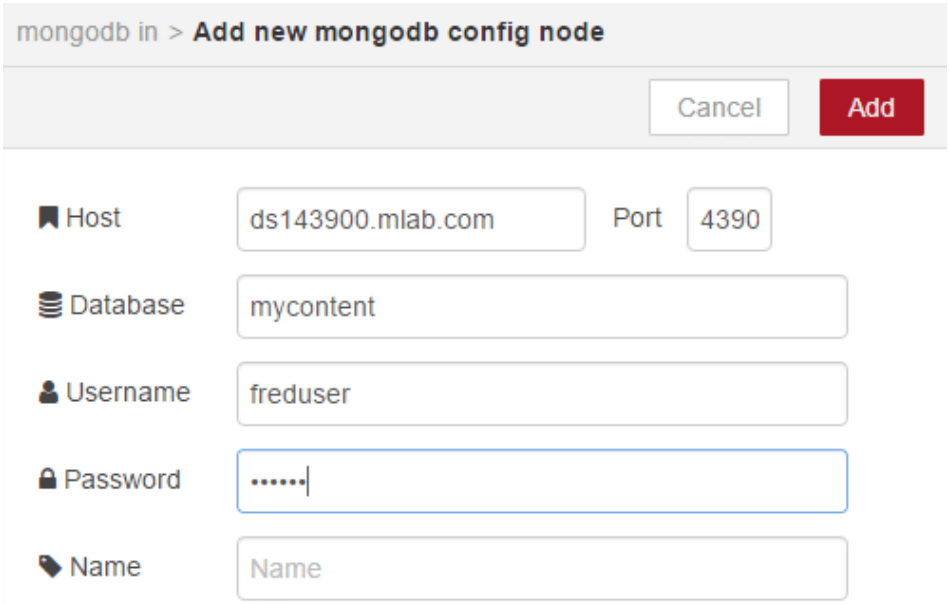
الآن، على Node-RED قم بربط العقد لكل من http-in وmongodb-in و template و http-out كما هو مبين بالشكل التالي:



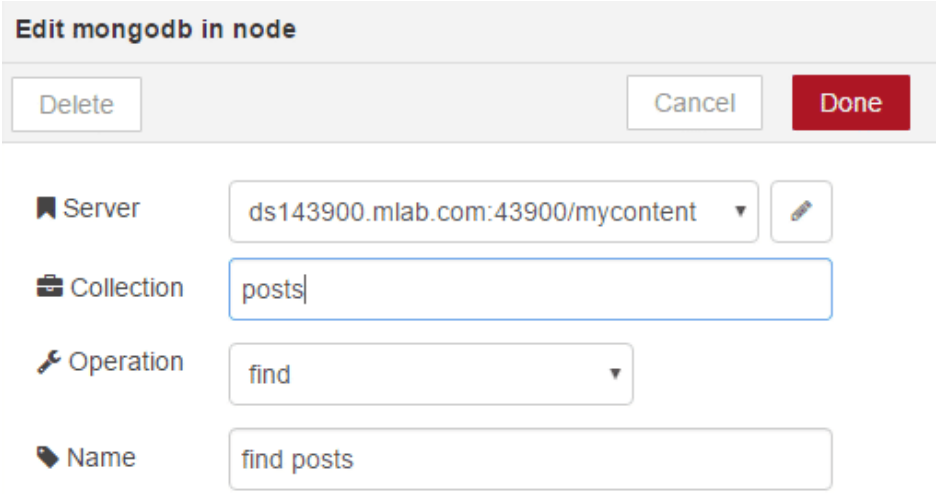
قم بالتعديل على عقدة http-in لقبول طلب GET على عنوان /public/posts/" URL" كما هو موضح بالصورة التالية :



سنقوم الآن بإعداد عقدة mongodb . انقر نقرا مذدوجا فوق العقدة وقم بإنشاء استصال خاد جديد. هنا هو المكان الذي سوف تستخدم المعلومات من mongolab الخاص بك :



انقر على إنشاء/تحديث وإعداد العقدة لاستخدام مجموعة "posts" التي أنشأتها. وإعداده للقيام بعملية العثور، والتي ثوف تجد جميع الوثائث في تلك المجموعة. سنطلق عليه اسم (find posts)) .



بمجرد إعداد عقد http و mongodb ، قم بتكوين عقدة html template للتعامل مع البيانات التي تم إرجاعها من قبل عقدة mongodb .

قم بإضافة التعليمات البرمجية التالية :

```
<!DOCTYPE html>
<html lang=”en”>
 <head>
 </head>
 <body>
 <h1>My Micro Blog</h1>
 <form action=”posts” method=”post”>
 <label for=”title”>Title</label>
 <input type=”text” class=”form-control” name=”title” placeholder=”Title…” />
 <label for=”post”>Post Content</label>
 <input type=”text” class=”form-control” name=”post” placeholder=”Say something…” />
 <button type=”submit” class=”btn btn-default”>Submit</button>
 </form>
 <div>
 {{#payload}}
 <div class=”post”>
 <h3>{{title}}</h3>
 <p>{{post}}</p>
 </div>
 {{/payload}}
 </div>
 </body>
</html>
```

هذا الكود، يستخدم Bootstrap لتصميم موقع الويب الخاص بك (http://getbootstrap.com/) وتوفير تخطيط استجابة. إذا قمت الآن بزيارة http://{your user name}.fred.senstecnic.com/api/public/posts ستكون قادر على رؤية الموقع الجديد الخاص بك .

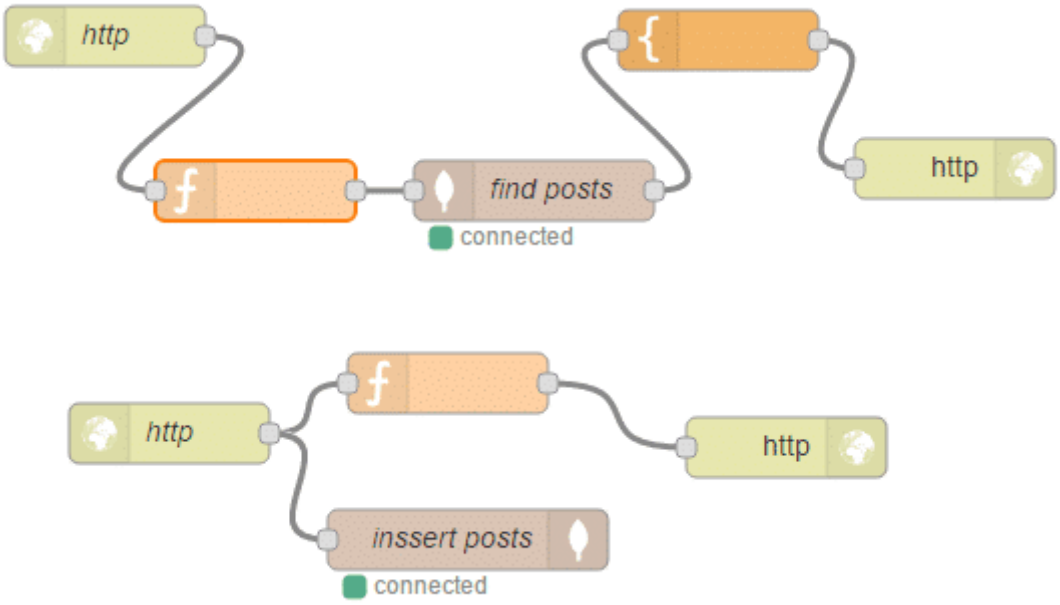
My Micro Blog

Title "Title..." Post Content "Say Submit

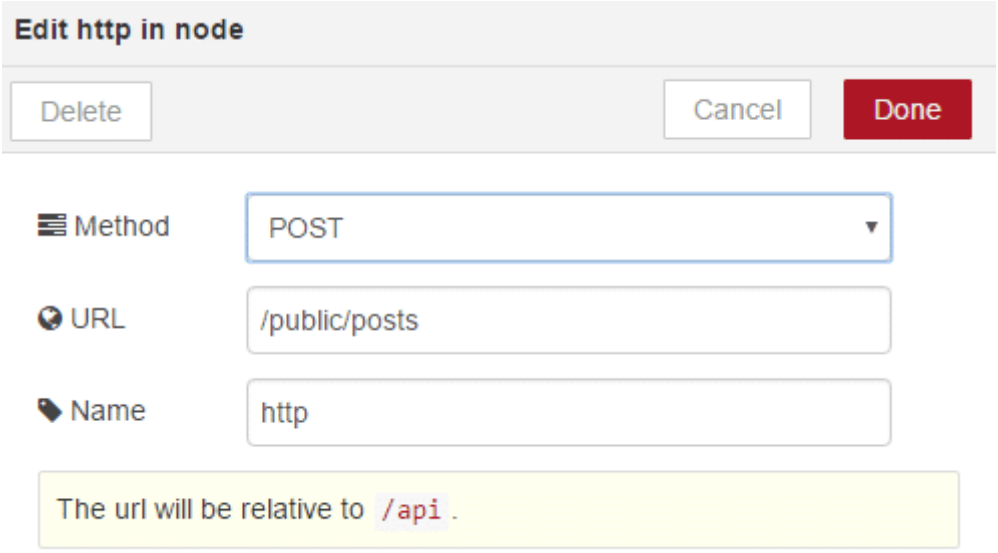
إذا تم ظهور خطأ ("MongoError: limit requires an integer")على لوحة الإخراج عند تبويب debug ، قم بإضافة عقدة function قبل عقدة mongodb وقت بكتابة الكود أدناه على عقدة function :

```
msg.limit = 5;
msg.skip = 0;
return msg;
```

حتى الآن، قمت بإنشاء التدفق الازم لعرض المشاركات.الآن تحتاج إلى إنشاء الجانب الأخر لخدمة blog، التدفق لإنشاء المشاركات وحفظها إلى مجموعة MongoDB الخاص بك. قم بتوصيل عقد http-in و mongodb-out و http-out كما هو موضح بالشكل التالي :



قم بالتعديل على عقدة http-in لقبول طلبات PUT في /public/posts/ .

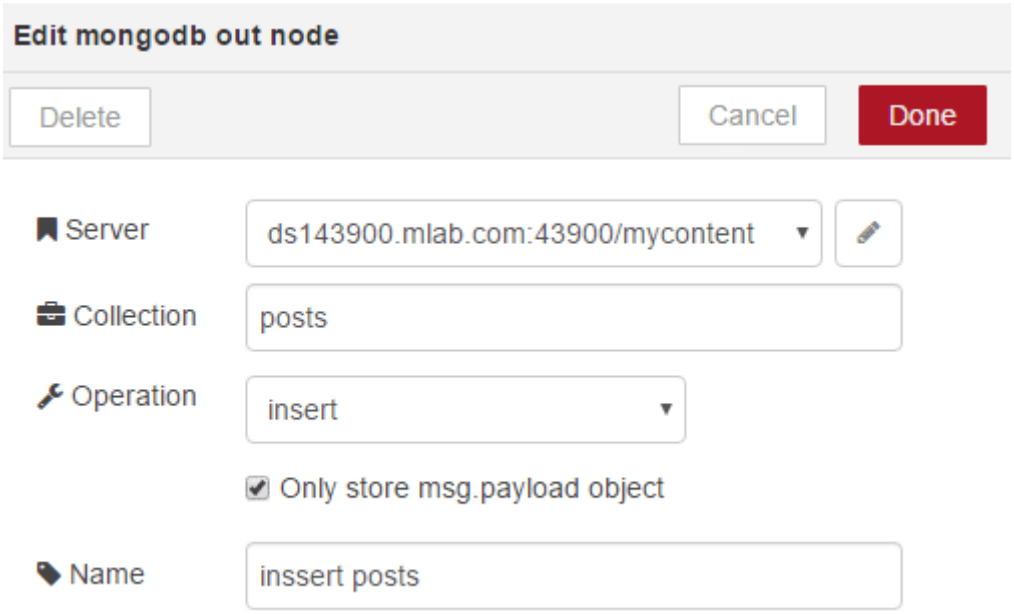


الآن سنقوم بتحرير عقدة function وإضافة التعليمات المبينه أدناه. حيث يبني http msg.header .

```
msg.headers = {
    "Location" : "https://{your username}.fred.sensetecnic.com/api/public/posts"
};
msg.statusCode = 302;
return msg;
```

لاحظ في السطر 2 تقوم بتعيين العنوان ليكون نقطة النهاية لخدمتك. سيستخدم هذا اسم المستخدم بدلا من 'guides' .

وأخيرا،سوف تحتاج إلى تحرير عقدة mongodb-out وحدد الخادم الذي تم تكوينه مسبقا. قم بتهيئة لاستخدام مجموعة "posts" و عملية إدراج "insert" . تأكد من فحص كائن msg.payload للتخزين فقط (only store msg.payload object) ، وبهذا سوف نتأكد من أننا فقط لتخزين البيانات في حمولة الرسالة وليس كائن الرسالة بأكملها .



الآن يمكنك الانتقال إلى عنوان url الخاص بك واستخدام موقع المدونة (blog) حيث سيظهر لك شكل ممثال للصورة أدناه :

My Micro Blog

Title Title... Post Content Say something... Submit

My First Post

Content from my first post