



مشاريع Node-RED متوسطة التدفق (الجزء الثاني)

المقدمة

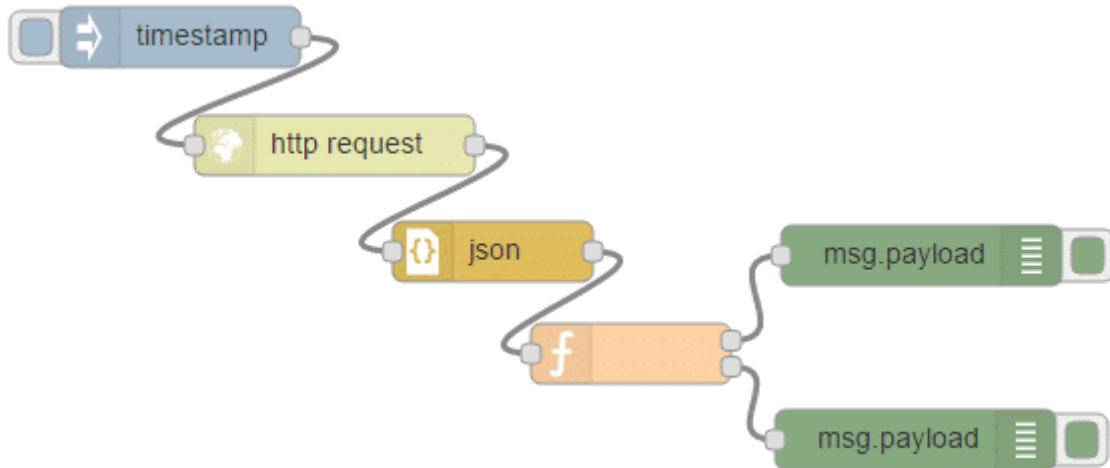
في هذا الدرس، سوف نبني على الأفكار التي تم عرضها في المحاضرة السابقة و التركيز على الأمثلة التي تكشف بعض المفاهيم الأساسية من الدرس السابق. و يشمل هذا الدرس مزيد من الأفكار مثل ارسال رسائل متعددة على مخرج واحد أو انشاء موقع مدونة خاص بك.

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 الزلازل التي يتم توفيرها من قبل المسح الجيولوجي الأمريكي (USGS).

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

أولاً، 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.

Edit http request node

Delete
Cancel
Done

Method
GET

URL
http://earthquake.usgs.gov/earthquakes/feed/v1.

☐ Enable secure (SSL/TLS) connection

☐ Use basic authentication

Return
a UTF-8 string

Name
Name

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

Edit function node

Delete
Cancel
Done

Name

Function

```

1 var outputMsgs = [];
2
3 for(i=0; i< msg.payload.features.length - 1 ; i++){
4     var feature = msg.payload.features[i];
5     outputMsgs.push(
6         {payload: { lat:feature.geometry.coordinates[1],
7                     lng:feature.geometry.coordinates[0],
8                     value:feature.properties.mag,
9                     message:feature.properties.place,
10                    timestamp:feature.properties.time
11                }
12            });
13 }
14
15 var msg2= {payload: "Second Output"}
16 return [ outputMsgs, msg2];

```

Outputs
2

See the Info tab for help writing functions.

شرح الكود :

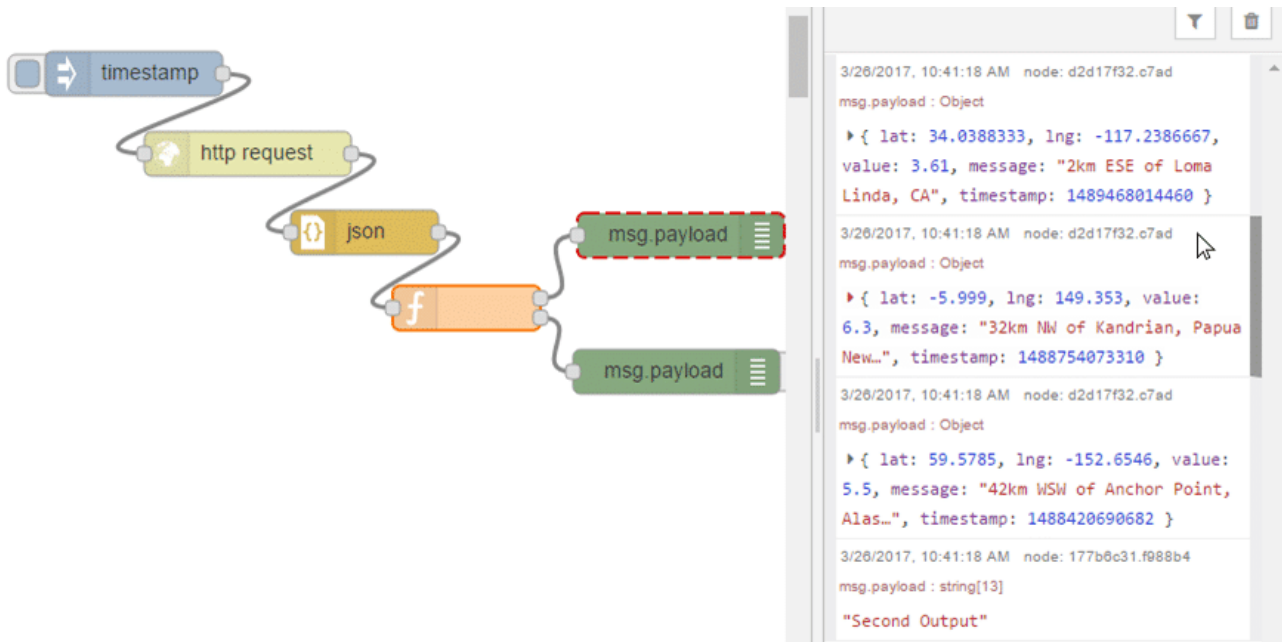
سيؤدي هذا الكود إلى إنشاء مصفوفة (outputMsgs) تحتوي على مصفوفة من الرسائل التي تم إنشاؤها من بياناتك. تمر حلقة التكرار (For loop) من خلال استجابة JSON وتقوم بإنشاء رسالة جديدة تحتوي على حمولة تحتوي على خط العرض (lat) ، وخط الطول

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

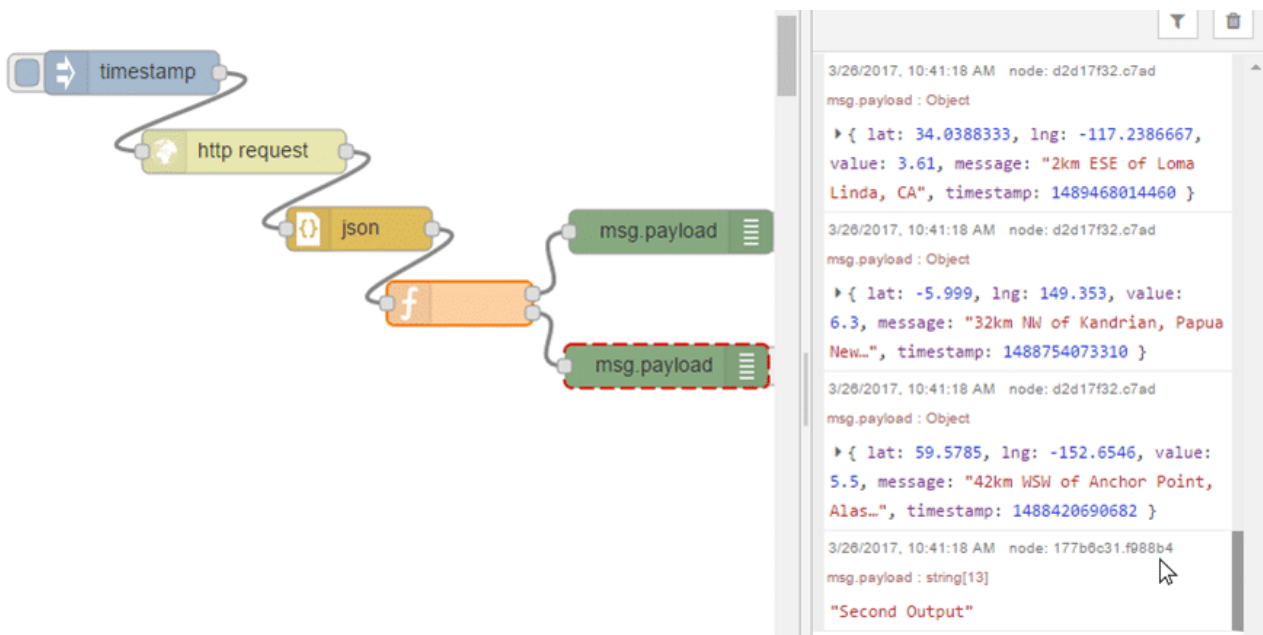
فإنه سيتم إنشاء متغير جديد يدعى (msg2) مع كائن رسالة جديدة تحتوي على عنصر الحمولة مع سلسلة "Second Output".

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 . لتناسب 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 .



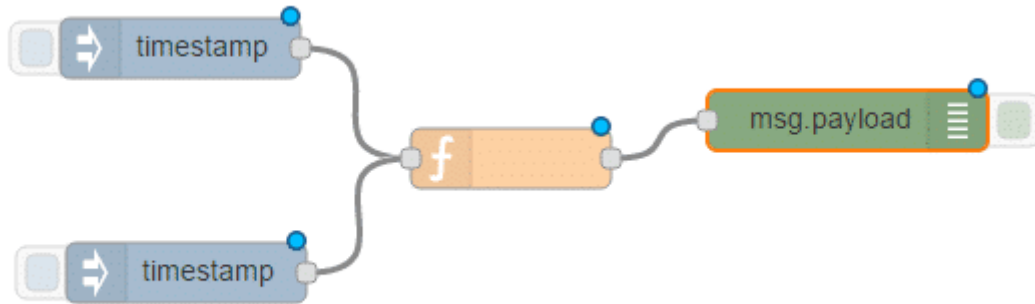
والمخرج الثاني يحتوي على رسالة واحدة (msg2) :



مدخلات متعددة على عقدة function :

تم تصميم عقد function في Node-RED لمعالجة الرسائل ككيانات واحدة. ومع ذلك، قد تَعمك 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 الشرط 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 :

Edit function node

Delete

Cancel

Done

Name

Ratio

Function

```
1 context.data = context.data || {};  
2 switch (msg.topic) {  
3   case "task1":  
4     context.data.task1 = msg.payload;  
5     msg = null;  
6     break;  
7   case "task2":  
8     context.data.task2 = msg.payload;  
9     msg = null;  
10    break;  
11    default:  
12      msg = null;  
13      break;  
14  }  
15  
16  if(context.data.task1 != null && context.data.task2 != null){  
17    var ratio = context.data.task1 / context.data.task2;  
18  }
```

Outputs

1

See the Info tab for help writing functions.

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

Edit inject node

Delete

Cancel

Done

Payload

a_z

3

Topic

task1

Repeat

none

☐ Inject once at start?

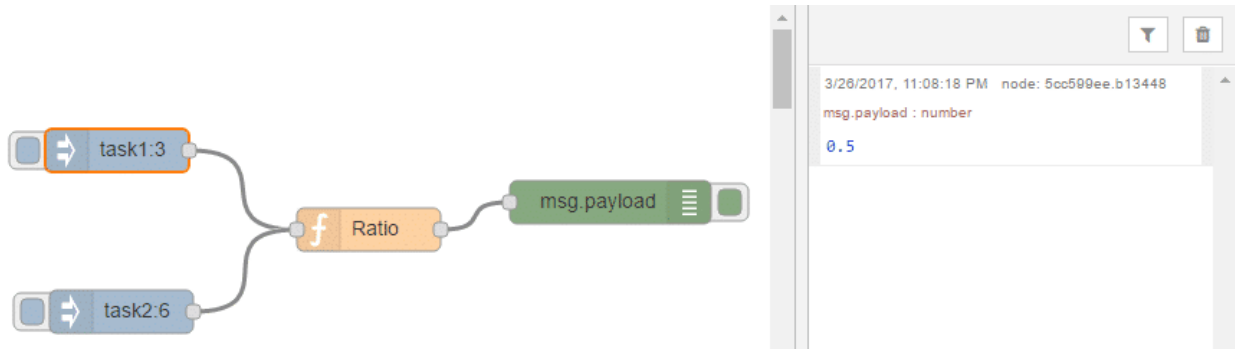
Name

Name

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

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

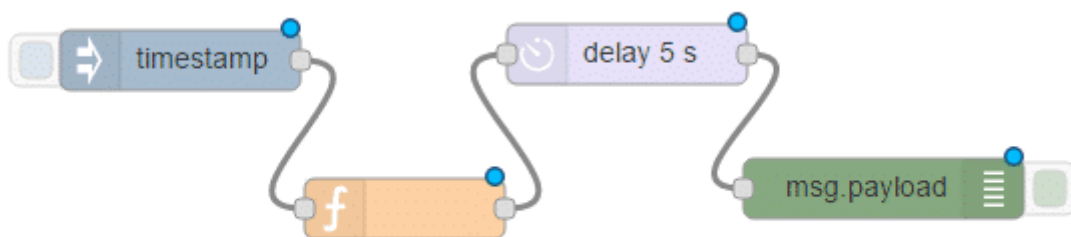


السماح لعقدة function بإرسال رسائل متعددة على مخرج واحد :

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 وإضافة 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 مخرجها 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 .

Edit function node

Delete
Cancel
Done

Name

Function

```

1 for (var i=0; i<5; i++){
2   node.send({payload:i});
3 }
4 return null;

```

Outputs

See the Info tab for help writing functions.

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

Edit delay node

Delete
Cancel
Done

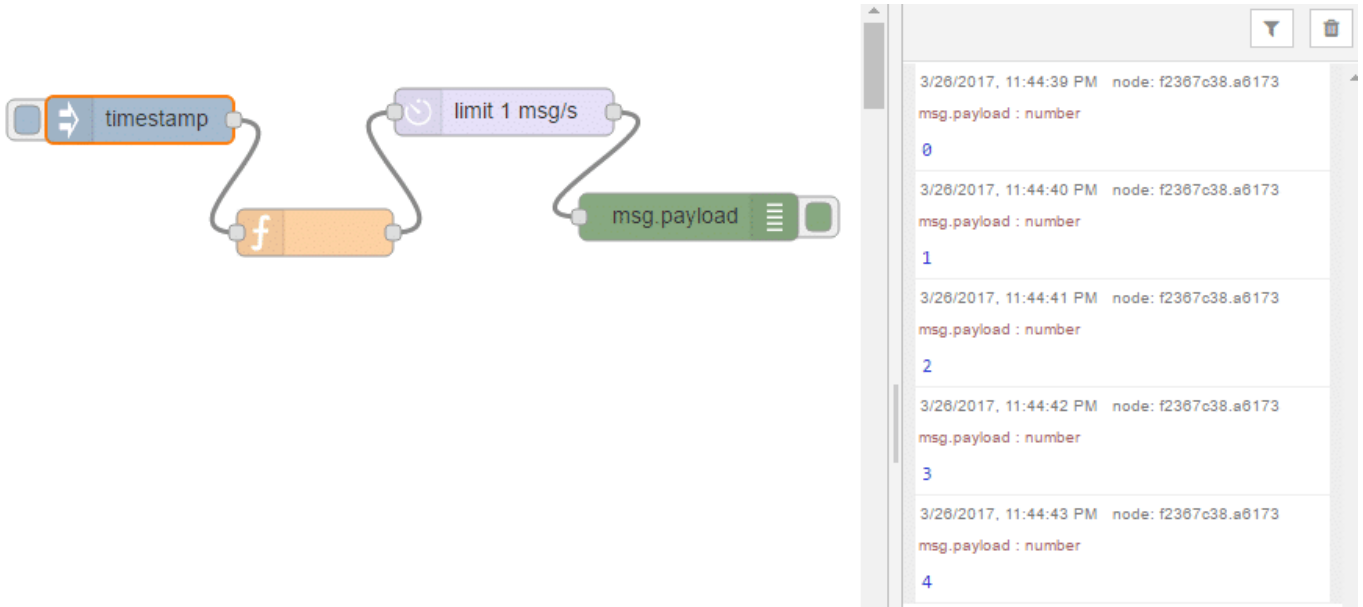
Action

Rate
 msg(s) per

☐ drop intermediate messages

Name

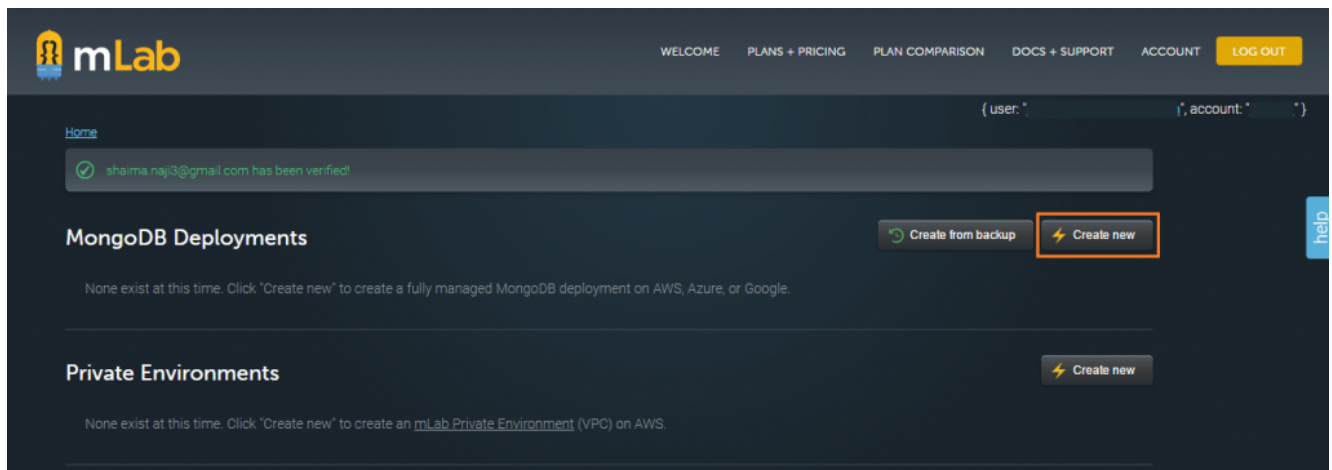
قم الآن بنشر التدفق، ثم الضغط على الزر الأيسر لعقدة inject وفحص لوحة الإخراج عند تبويب debug ، ستلاحظ وصول الرسائل كل ثانية، انظر الشكل التالي :




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 باستخدام عقد MongoDB كتخزين للمشاركات، وعقد http لتوفير نقاط النهاية للخدمة وعقدة html لتنسيق صفحة ويب للمدونة.

سنقوم باستخدام 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 .


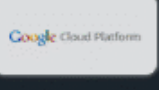
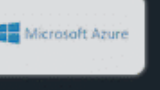


[Home](#)

Create new deployment

Fill out this form to create a brand-new MongoDB deployment in the cloud location of your choice. Alternatively, you can [create a new deployment from backup](#)

Cloud Provider:

Region: **Amazon's US East (Virginia) Region (us-east-1)**

Plan ([view plan details](#)):

Single-node

Replica set cluster

These plan(s) are perfect for development/testing/staging environments as well as for utility instances that do not require high availability.

Standard Line

The most economical plans for applications running on AWS.

<input checked="" type="radio"/> Sandbox (shared, 0.5 GB)	FREE
<input type="radio"/> M3 Single-node (7.5 GB, 120 GB SSD block storage)	\$ 420
<input type="radio"/> M4 Single-node (15 GB, 240 GB SSD block storage)	\$ 835
<input type="radio"/> M5 Single-node (34.2 GB, 480 GB SSD block storage)	\$ 1310
<input type="radio"/> M6 Single-node (68.4 GB, 700 GB SSD block storage)	\$ 2045

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

Database Name:

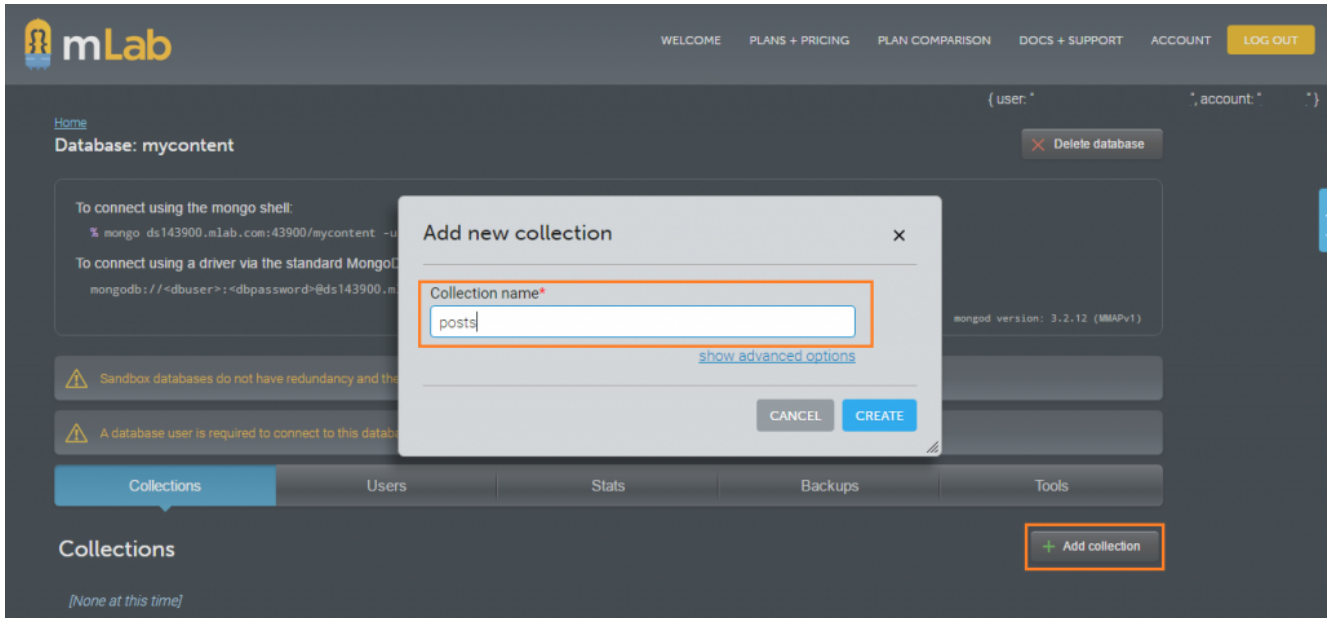
mycontent

Price:

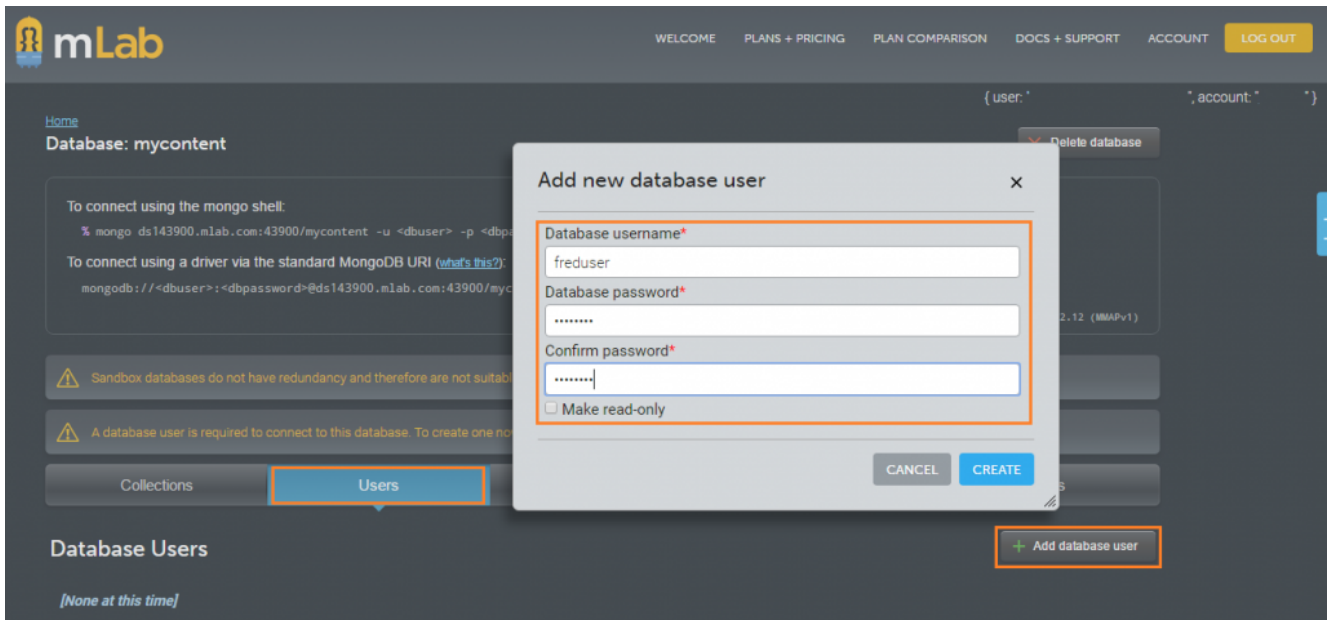
\$0 / month

Create new MongoDB deployment

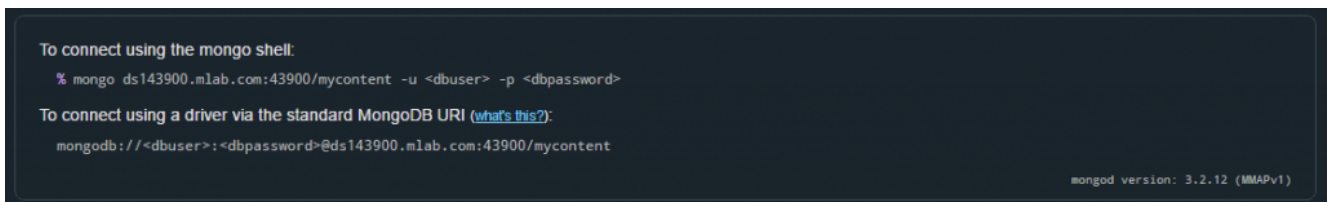
يسمح لك MongoDB بإنشاء مجموعات "collections" لكل قاعدة بيانات. وهي مماثلة للجداول في قواعد البيانات العلائقية، كل مجموعة "collection" تحمل الوثائق "documents" التناظرية إلى السجلات "records" في قواعد البيانات العلائقية. من أجل البدء في استخدام قاعدة البيانات الخاصة بك تحتاج إلى إنشاء مجموعة جديدة. حدد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



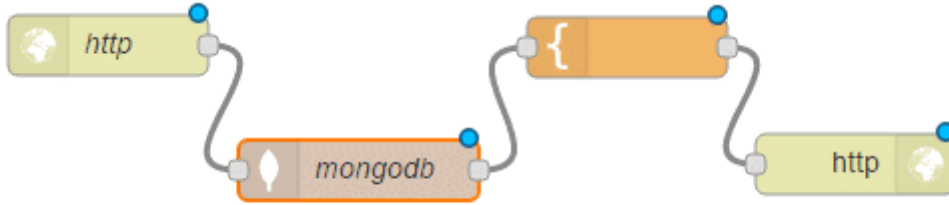
وأخيراً، سوف تحتاج مستخدم للإتصال بقاعدة البيانات تلك. قم بإضافة مستخدم قاعدة بيانات جديدة. وسوف نقوم بتسميتها "freduser".



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



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



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

Edit http in node

Delete
Cancel
Done

Method
GET

URL
/public/posts

Name
Name

The url will be relative to /api .

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

mongodb in > Add new mongodb config node

Cancel
Add

Host
ds143900.mlab.com
Port
4390

Database
mycontent

Username
freduser

Password
.....

Name
Name

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

Edit mongodb in node

Delete
Cancel
Done

Server
ds143900.mlab.com:43900/mycontent

Collection
posts

Operation
find

Name
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>
    </div>
    {{/payload}}
  </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 .

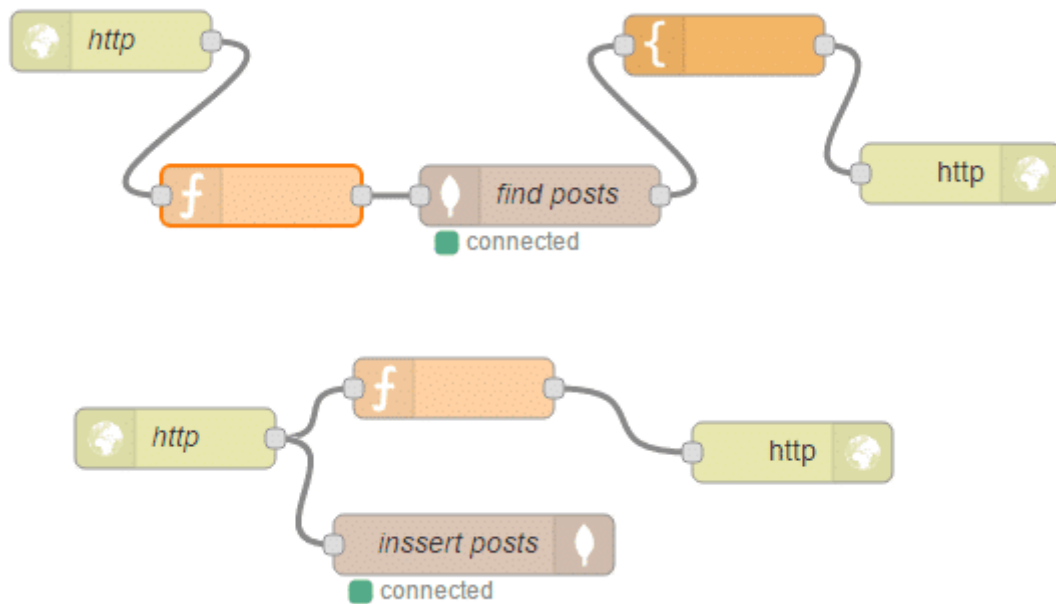
My Micro Blog

Title Post Content

إذا تم ظهور خطأ ("MongoError: limit requires an integer") على لوحة الإخراج عند تبويب debug ، قم بإضافة عقدة function قبل عقدة mongodb ووقت 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 الآخر 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/ .

Edit http in node

Delete
Cancel
Done

Method
POST

URL
/public/posts

Name
http

The url will be relative to `/api` .

الآن سنقوم بتحرير عقدة function وإضافة التعليمات المبينه أدناه. حيث يبنّي 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 .

Edit mongodb out node

Delete
Cancel
Done

Server
ds143900.mlab.com:43900/mycontent

Collection
posts

Operation
insert

☒ Only store msg.payload object

Name
insert posts

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

My Micro Blog

Title Post Content

My First Post

Content from my first post