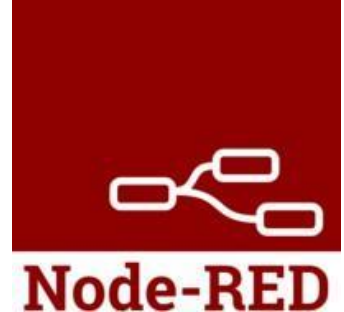


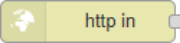

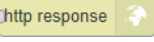
Laboratory 3



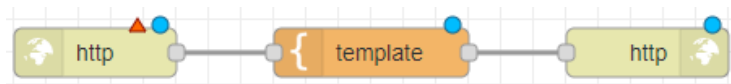
Full-stack application

First, let's create a front end with the classic "Hello world" message.

We are going to use the following nodes:

- http in 
- template 
- http response 

The flow so far should look like this:



Edit the URL of the http in node to the desired page name. In my case, I chose IIOTCA.

Method	GET
URL	/IIOTCA

In the template, you can choose the syntax highlight as HTML, and then just copy this basic HTML document example:

```
<!DOCTYPE html>
<html>
<body>

<h1>Hello world</h1>
<p>This is the first example.</p>

</body>
</html>
```

Save and run the flow. Then go to localhost:1880/*page name*

In case you run it on IBM, go to your App URL/*page name*

Details

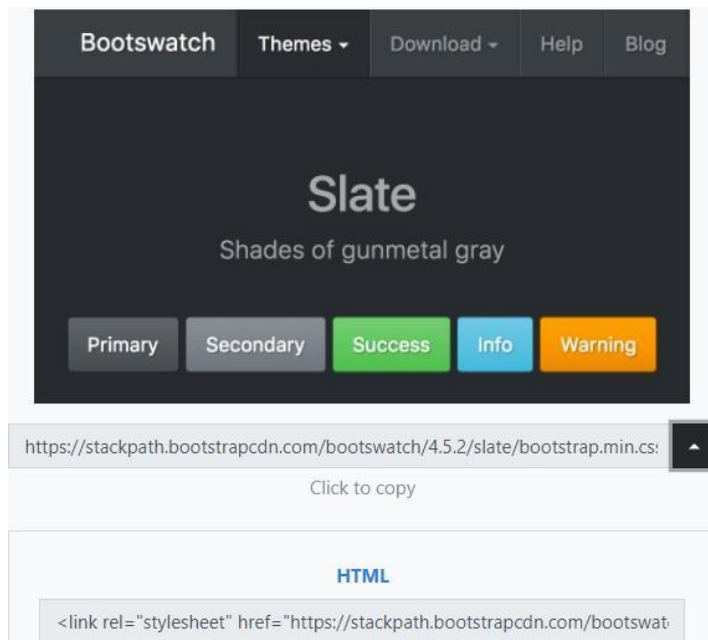
App URL

<https://iiotca-lab.eu-gb.mybluemix.net>

You should have the following output.



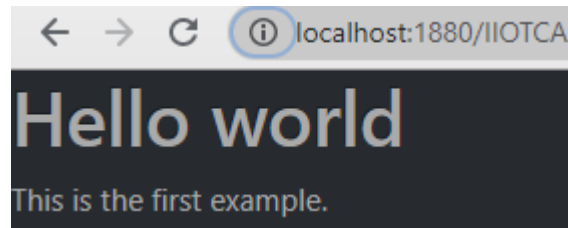
Let's edit the page a bit... We are going to add styles with CSS. For this, we can use a bootstrap template. I am going to use BootstrapCDN. CDNs are content delivery networks, which lets the users load CSS, Js and images remotely, from their servers. <https://www.bootstrapcdn.com/bootswatch>



Choose a template, and copy the html link. Then create a `<head>` link `</head>` structure inside the html block:

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="https://s
  </head>
</body>
```

If you want, you can also delete the “https:”, and continue from the two slashes. You can then once again check the output on the site:

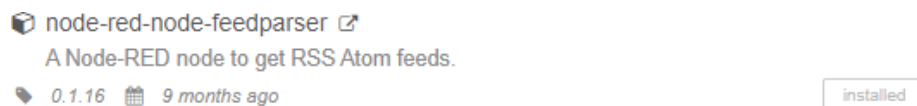


We can make another example of online media monitoring, by reading RSS from online news sites. RSS is a web feed that allows users and applications to access updates to websites in a standardized, computer-readable format.

After reading the information, we can also filter it by detecting different keywords.

How are we going to read the data? Well, RSS feeds are based on XML, and it gives you direct access to Headlines, Timestamps, Abstract and Link.

Let's move on to the Node-RED part. First, we have to add a feed parser node. Firstly, we have to install it:



The feed parser node is simple to use, you just have to feed it a link towards a RSS feed and the refresh rate (when testing, you can set this to 0).

Here are some examples of news RSS feeds. You can choose whichever you want. https://blog.feedspot.com/world_news_rss_feeds/

When you open the feed, it should have a .xml ending for the page:

`feeds.bbc.co.uk/news/world/rss.xml`

And the content should look as follows:

Coronavirus: WHO criticises EU over vaccine export controls

"Vaccine nationalism" risks causing a "protracted recovery", the organisation's chief warns.

Covid: EU approves AstraZeneca vaccine amid supply row

The EU approves AstraZeneca's Covid vaccine, but accuses the firm of breaking supply commitments.

OR This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:media="http://search.yahoo.com/mrss/" xmlns:atom="http://www.w3.org/2005/Atom">
  <channel>
    <title>NYT > World News</title>
    <link>https://www.nytimes.com/section/world</link>
    <atom:link href="https://-.cloudfunctions.net/?" rel="self" type="application/rss+xml"/>
    <description/>
    <language>en-us</language>
    <copyright>Copyright 2021 The New York Times Company</copyright>
    <lastBuildDate>Sat, 30 Jan 2021 09:42:13 +0000</lastBuildDate>
    <pubDate>Sat, 30 Jan 2021 09:42:13 +0000</pubDate>
  </channel>
</rss>
```

Once you have the link, paste it into your node:

 Feed url

 Refresh minutes

 Name

Add a debug node showing the complete object, and run it:









In the output message we find an article object, which contains the title, description, summary etc.

```
1/30/2021, 11:49:46 AM node: b6d825ac.b1b378
https://www.bbc.co.uk/news/world-europe-55863069 :
msg : Object
  object
    topic:
      "https://www.bbc.co.uk/news/world-europe-55863069"
    payload: "Only essential travel from outside the bloc will be allowed from Sunday, but a lockdown is resisted."
    article: object
      _msgid: "f3c1797c.3c6d48"
```

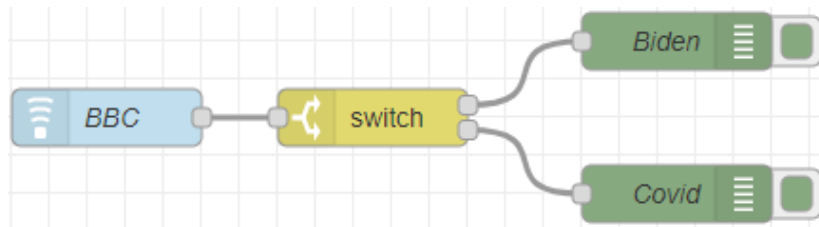
Let's filter the news. For example, we may only want to see news about Covid, but someone else wants news about Biden. For this, we can use a switch node. Using it, we check if Covid or Biden appear in the description. To access the description, we use `msg.article.description`. (We can also use regex for this)

Set the node up to check if it contains the searched words:

Property

	<input type="text" value="contains"/>		<input type="text" value="Biden"/>	→ 1	
	<input type="text" value="contains"/>		<input type="text" value="Covid"/>	→ 2	

Make two separate output nodes, one for Biden news and one for Covid.

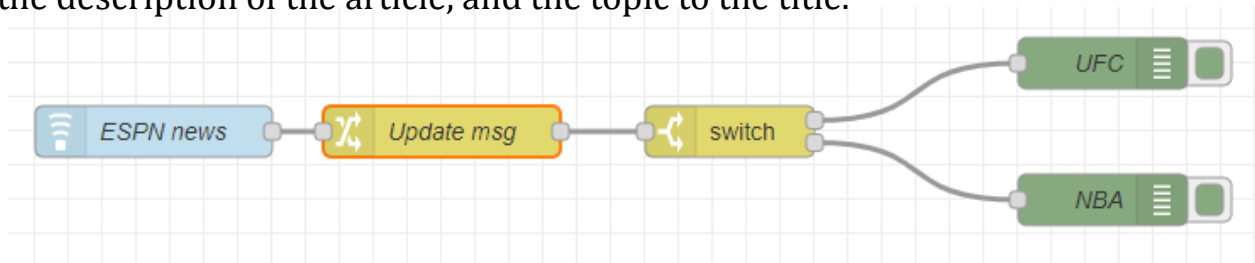


When we deploy it, we can see that the news are separated into the different areas of interest of the users:

```
1/30/2021, 11:55:25 AM node: Covid
https://www.bbc.co.uk/news/world-latin-america-55829424 : msg : Object
{
  topic: "https://www.bbc.co.uk/news/wor...",
  payload: "Some Latin American countries ...",
  article: object,
  _msgid: "ae51f493.6e9488"
}

1/30/2021, 11:55:25 AM node: Biden
https://www.bbc.co.uk/news/world-us-canada-55850498 : msg : Object
{
  topic: "https://www.bbc.co.uk/news/wor...",
  payload: "Activists celebrated Biden's r...",
  article: object,
  _msgid: "94fccdc9.346dd"
}
```

To make it easier to access, we can filter the wanted information using a change node. Just like we did in the previous lab, we can set the msg.payload and topic to our areas of interest. In this case, we can set the msg.payload to the description of the article, and the topic to the title.



Rules

Set	▼	▼ msg. topic	×
	to	▼ msg. article.title	
Set	▼	▼ msg. payload	×
	to	▼ msg. article.description	

By doing this, our output should look something like this.

```
1/30/2021, 11:59:52 AM node: Covid
Coronavirus: What's behind Latin America's oxygen
shortages?: msg : Object
  object
    topic: "Coronavirus: What's behind
Latin America's oxygen shortages?"
    payload: "Some Latin American
countries struggle with
insufficient oxygen as Covid-19
continues to spread."
```

But what if we want to also include the link in the payload, alongside the description? We can do this using a template node with the following syntax (we need 3x { for formatting):

```
Property      msg.payload
Template      Syntax Highlight: mustache
1  {{{article.description}}}
2
3  Read more: {{{article.link}}}
```

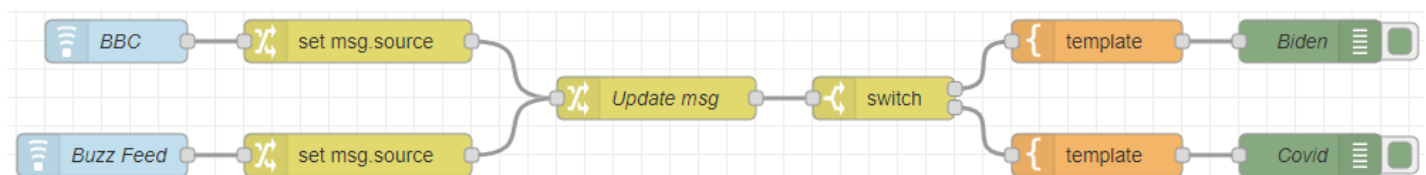
If we use this, remove the payload setting from the change node. Finally, the new output should look like this:

```
object
  topic: "Covid: EU approves
AstraZeneca vaccine amid supply
row"
  payload: string
    The EU approves AstraZeneca's Co
vid vaccine, but accuses the fir
m of breaking supply commitme
nts.

    Read more: https://www.bbc.co.u
k/news/world-europe-55862233
```

If you want, you can also add a second news site. If you do this, you can also add a change node, to set a source parameter.

```
Set      msg.source
to      a_z BBC
```



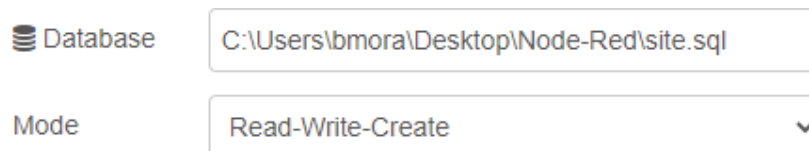
Back-end

First things first, we have to make sure we have sqlite nodes installed:



Make a new flow, and if you want, I recommend having all of the flows named: Site, RSS, DB etc.

Add the sqlite node and in the properties section set the path where you want your data to be stored. Leave the SQL query unchanged.



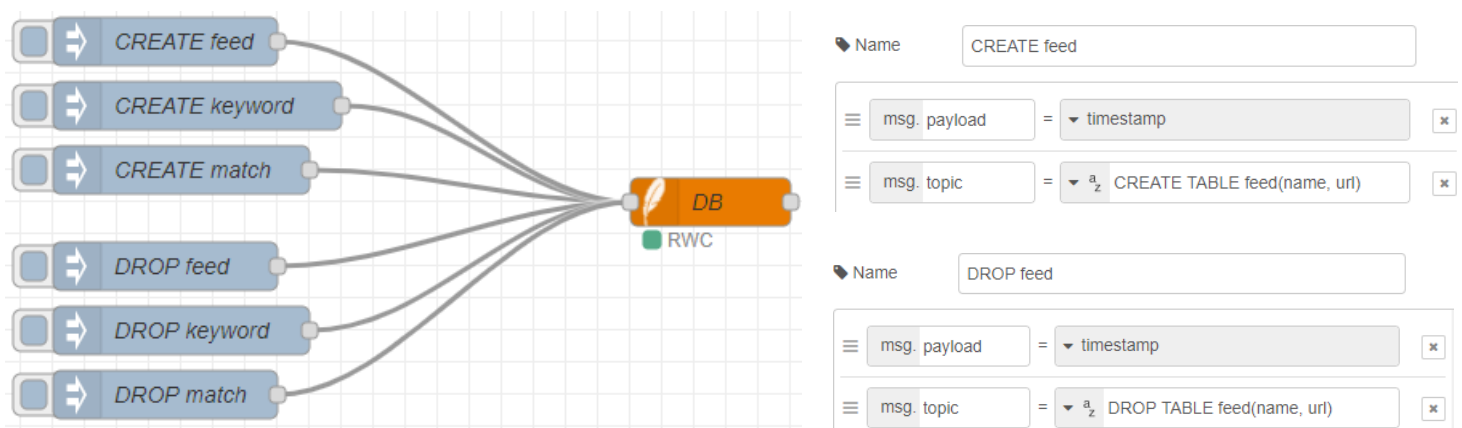
Add three inject nodes with the following topics, and timestamp as the payload:

```
CREATE TABLE feed(name, url)
```

```
CREATE TABLE keyword(expression)
```

```
CREATE TABLE match(link, title, description, UNIQUE(link))
```

Also, name the 3 nodes accordingly, like Create feed, Create keyword, Create match. Afterwards, copy those 3 nodes, and make them drop the database (simply replace CREATE with DROP everywhere).



So, to sum up the three tables:

- Feed will contain the name of the site the news were extracted from, and url will be the link towards the RSS feed
- Keyword will contain the words we want to search for
- Match will contain the title, description and link of the extracted news

Let's go back to the site flow. This is where we are going to manage our feeds. Edit the URL from the first node to IIOTCA/feed.

Let's add a table on the respective page. For this, go back to the bootstrap page from before, open the page that you chose, and go to the category TABLES, and expand it by clicking on the button from the bottom right:

Tables				
Type	Column heading	Column heading	Column heading	< >
Active	Column content	Column content	Column content	
Default	Column content	Column content	Column content	

Copy the code and put it instead of the two paragraphs from before.

```
<body>
<table class="table table-hover"><table>
|
</body>
```

Below I will attach a html file, you can find the structures for the template node inside... follow the steps and extract the structure one step at a time, or you will get a lot of errors.

<https://github.com/bmorariu/IIOTCA/blob/main/feed.html>

Deploy the changes and check the web page:

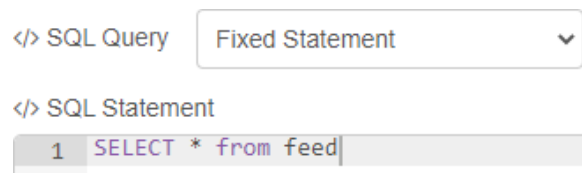
localhost:1880/IIOTCA/feed	
TYPE	COLUMN HEADING
ACTIVE	Column content
DEFAULT	Column content

Since we have too many rows, we can delete most of them. Also, we can put the table class inside a div.

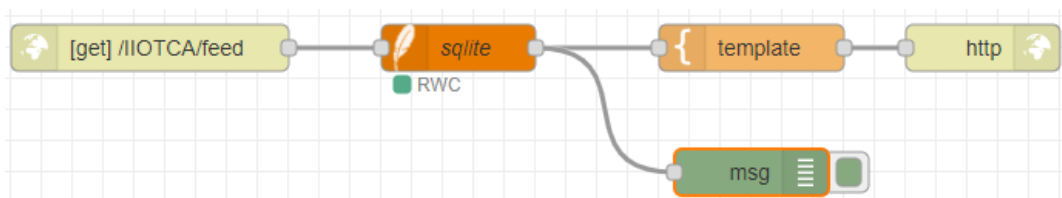
Add the container div class, and in the table, only keep the following:

```
<div class = "container">
  <table class="table table-hover">
    <thead>
      <tr>
        <th scope="col">Name</th>
        <th scope="col">Url</th>
        <th scope="col"></th>
      </tr>
    </thead>
    <tbody>
      <tr class="table-primary">
        <td>Column content</td>
        <td>Column content</td>
        <td>Column content</td>
      </tr>
    </tbody>
  </table>
</div>
```

Add a sqlite node with a fixed statement as a query.



Add a debug node outputting the complete message object from the database, to see the output. After all those changes, your flow should now look like this:



You can refresh the feed webpage and you will see the output:

```
1/27/2021, 5:45:39 PM node: 5cf6b48c.152c2c
msg : Object
  object
    _msgid: "9b568c06.8bb78"
    payload: array[0]
    req: object
    res: object
```

We see that the output comes as an array, which is empty, because we currently have no information in the table. For this, we should use a new http node, but this time, the method should be POST.

Afterwards, we need to update the template node to include a form. Go to the bootstrap site and to the form source code and find a text input, or you

```
<div class="form-group">
  <label for="exampleInputEmail1">Email address</label>
  <input type="email" class="form-control" id="exampleIr
```

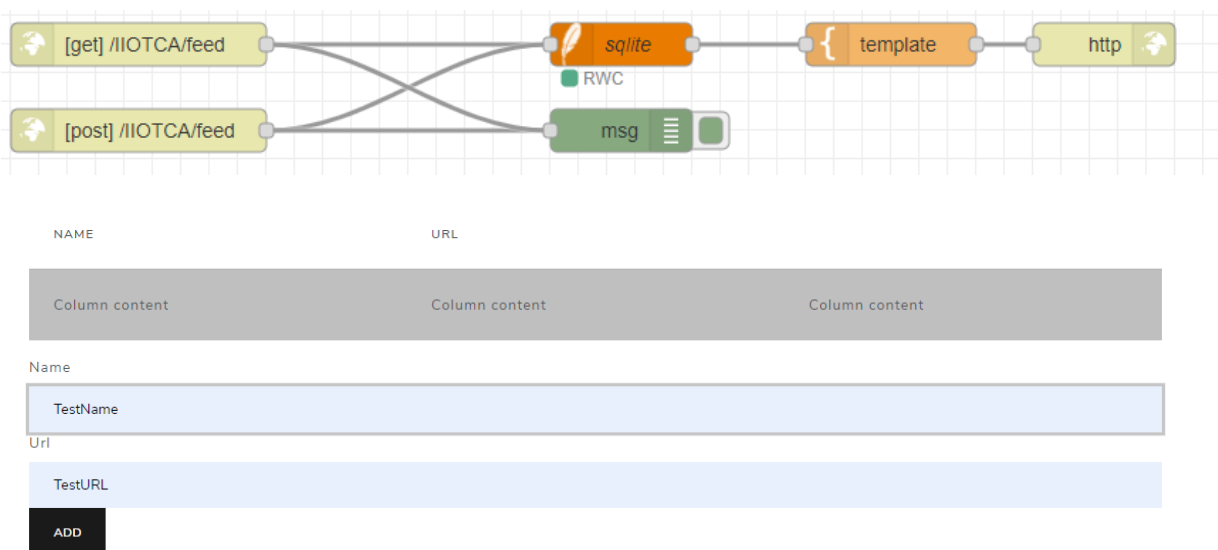
can use the following:

```
<form method="POST" action="/IIOTCA/feed">
  <label for="name">Name</label>
  <input class="form-control" id="name" name="name">
  <label for="url">Url</label>
  <input class="form-control" id="url" name="url">
</form>
```

Then you can check the site again. What we need to finish the form is a button. For this, go to the bootstrap page and get the code for a button, or just add this right before you close the form.

```
<button type="submit" class="btn btn-primary">Add</button>
```

Firstly, link the post with the output node, and then try to post an example:



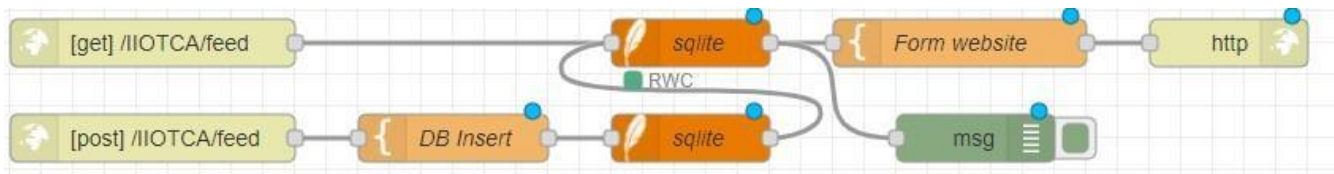
In the debug window, we can see that the payload contains the name and URL previously inserted:

```
1/27/2021, 10:07:02 PM node: 5cf6b48c.152c2c
msg: Object
  object
    _msgid: "c897afc0.e3389"
    payload: object
      name: "TestName"
      url: "TestURL"
    req: object
    res: object
```

Then, we need to insert this information into the database. For this we can use a simple template node with the property being msg.topic and with following content:

```
INSERT INTO feed VALUES ('{{payload.name}}','{{payload.url}}')
```

To insert it, we use another sqlite node, but this time we send the topic. Then, we have quite a bit of re-wiring to do:



Test the form and the output again. This time, the array indexes should increase:

```
1/27/2021, 10:38:03 PM node: 5cf6b48c.152c2c
INSERT INTO feed VALUES ('TestName','TestURL') :
msg: Object
  object
    _msgid: "e86ee9ae.29d498"
    payload: array[1]
      0: object
        name: "TestName"
        url: "TestURL"
    topic: "INSERT INTO feed VALUES ('TestName','TestURL')"
    req: object
    res: object
```

To be able to see the inputs in the table, we need to update the template node. For this, add `{{#payload}}` and `{{/payload}}` inside `<tbody>`.

```
<tbody>
  {{#payload}}
  <tr class="table-primary">
    <td>Column content</td>
    <td>Column content</td>
    <td>Column content</td>
  </tr>
  {{/payload}}
</tbody>
```

To be able to delete those from the table, we need to add a delete node. You can get any from the bootstrap, or just use the following code between the two payloads previously written:

```
<tr class="table-primary">

  <td>{{name}}</td>

  <td>{{url}}</td>

  <td><a href="/IIOTCA/feed/delete/{{rowid}}" class="btn btn-danger">delete</a></td>

</tr>
```

To be able to delete rows, we first have to configure it. Go to the sqlite node with a fixed statement and replace the current statement with

```
</ SQL Query  Fixed Statement  v>

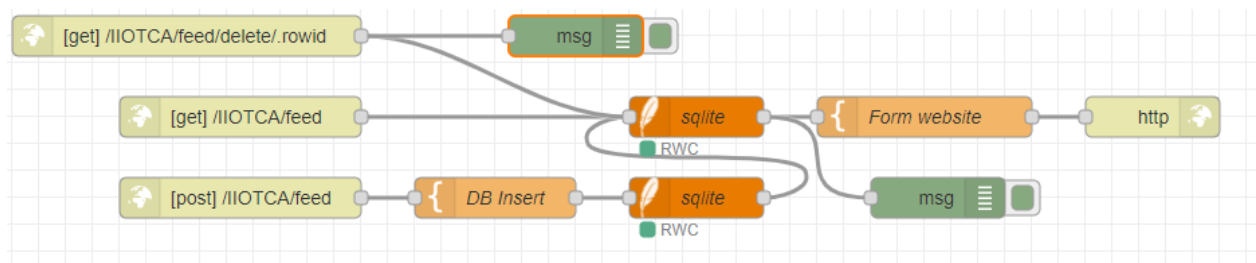
</ SQL Statement

1 SELECT rowid, * FROM feed
```

To be able to delete it, we also need a new http node with a get method, where the url is

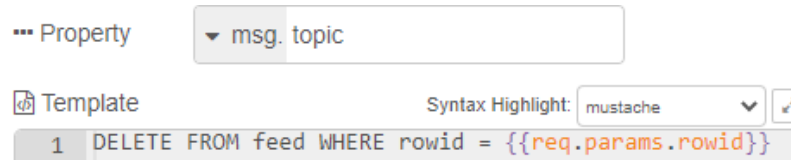
`/IIOTCA/feed/delete/:rowid`

Before trying to delete it, add an output node which displays the complete message object after the delete node. When we try to delete something from the table, we can see what we need to update:

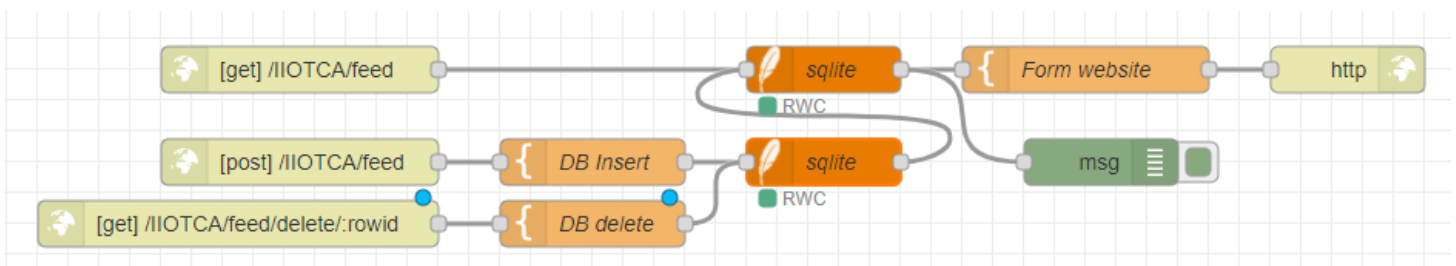


In the output window we can find the rowid in req.params.rowid

We add a new template node, where we just delete the recordings from feed for the given rowid.



Then, we simply link the template node to the sqlite node with the message topic. You can also rearrange the nodes to not overload the wiring:



After this, everything should be set up correctly. You should be able to add and delete the information stored in the database.

After you make sure both adding and deleting work, prepare to add the two(or more) sources you have previously chosen. For this, we have to tell mustache not to encode the html entities. We do this by adding another set of curly braces in the DB insert area.

NAME	URL	
ESPN	https://www.espn.com/espn/rss/news	DELETE
FOX	https://api.foxsports.com/v1/rss?partnerKey=zBaFxRyGKCfxBagJG9b8pqLyndmvo7UU	DELETE

Two are still not that many sources, so let's add 2-3 more. (from https://blog.feedspot.com/world_news_rss_feeds/).

The sizes of the input areas are way too big, and of course we care about design. To do this, go and edit the template node. Add the following inside `<div class = "container">`.

```
<div class="row">
  <div class="col-md-6">

</div>
<div class="col-md-6">

</div>
</div>
</div>
```

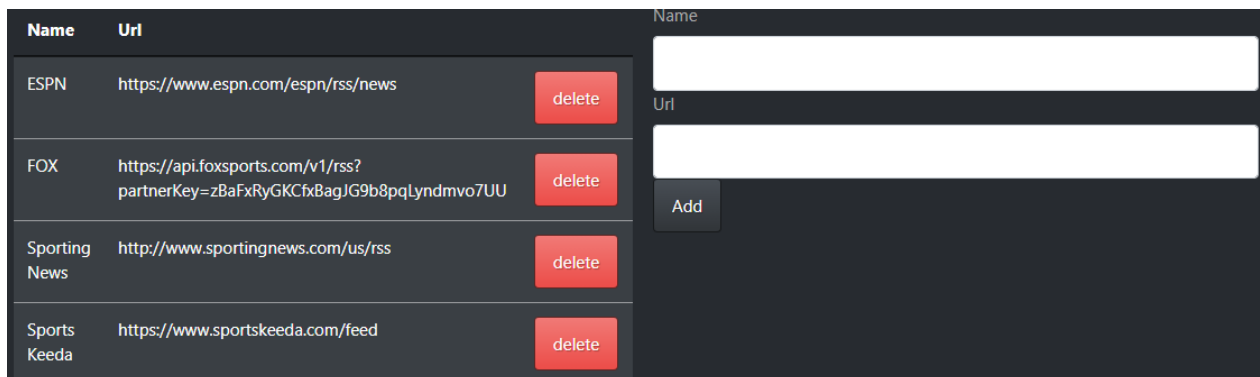
```
<div class = "container">
  <div class="row">
    <div class="col-md-6">

    </div>
    <div class="col-md-6">

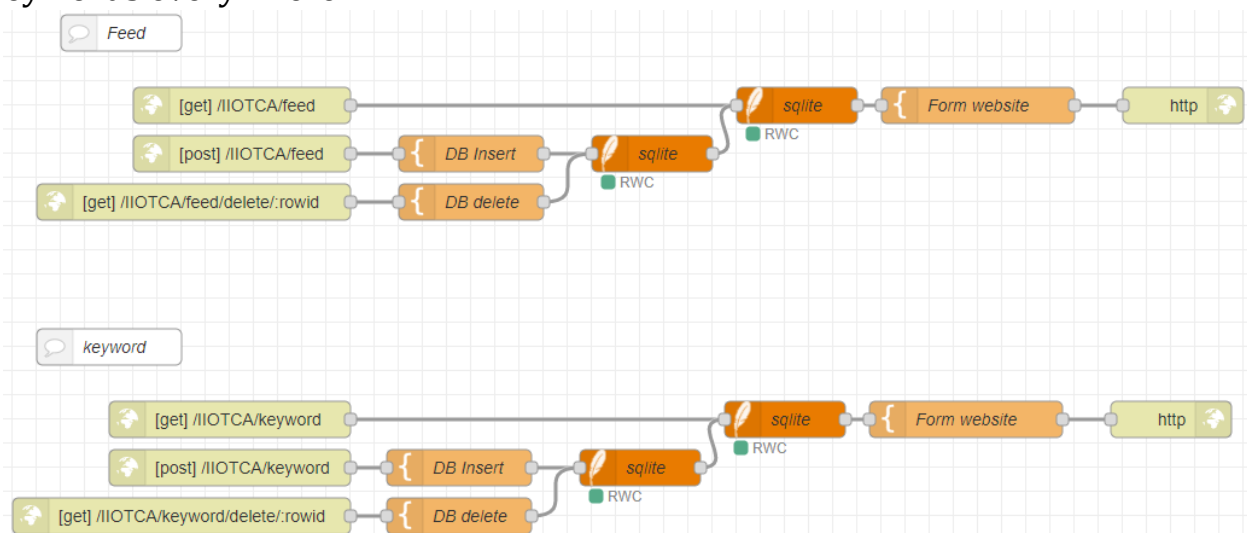
    </div>
  </div>
  <table class="table table-hover">
```

In the first gap, we are going to put the entire table structure, and in the second one we put the form.

In some cases, the sizes are gonna be off, in that case, just switch the bootstrap you used in the stylesheet with a new one. (P.S. At this point, you can copy the entire structure from the html if you need help with it)



Now this page is done, we can move on to the next one(keyword). Go to the site flow, and make a copy of the existing nodes and replace feed with keywords everywhere.



After that, edit the template node containing the html structure. We need to change the name column to expression, and delete the URL column. Do the same changes in the table area and the form. After that, do the same for the insert command.

```
<div class="row">
  <div class="col-md-6">
    <table class="table">
      <thead>
        <tr>
          <th scope="col">Expression</th>
          <th scope="col"></th>
        </tr>
      </thead>
      <tbody>
        {{#payload}}
        <tr class="table-primary">
          <td>{{expression}}</td>
          <td><a href="/IIOTCA/keyword/delete/{{rowid}}" c1
        </tr>
        {{/payload}}
      </tbody>
    </table>
  </div>
  <div class="col-md-6">
    <form method="POST" action="/IIOTCA/keyword">
      <label for="expression">Expression</label>
      <input class="form-control" id="expression" name="e
      <button type="submit" class="btn btn-primary">Add</
    </form>
  </div>
</div>
```

Now deploy the changes and go to the keyword page. Add the keywords we added before.

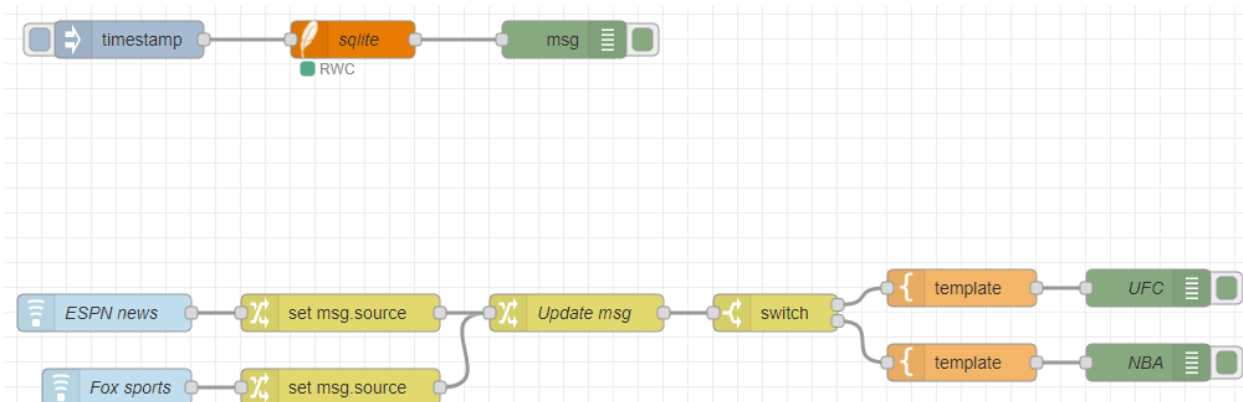
Go back to the RSS reader. Move the previous flow down to make room for a new one.

Add an inject node, followed by a DB node. You can copy the fixed statement one from the site. Link them together, and add a debug node to check the output.

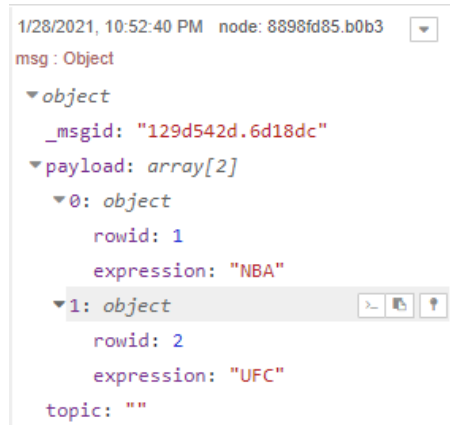
</> SQL Query Fixed Statement

</> SQL Statement

```
1 SELECT rowid, * FROM keyword
```

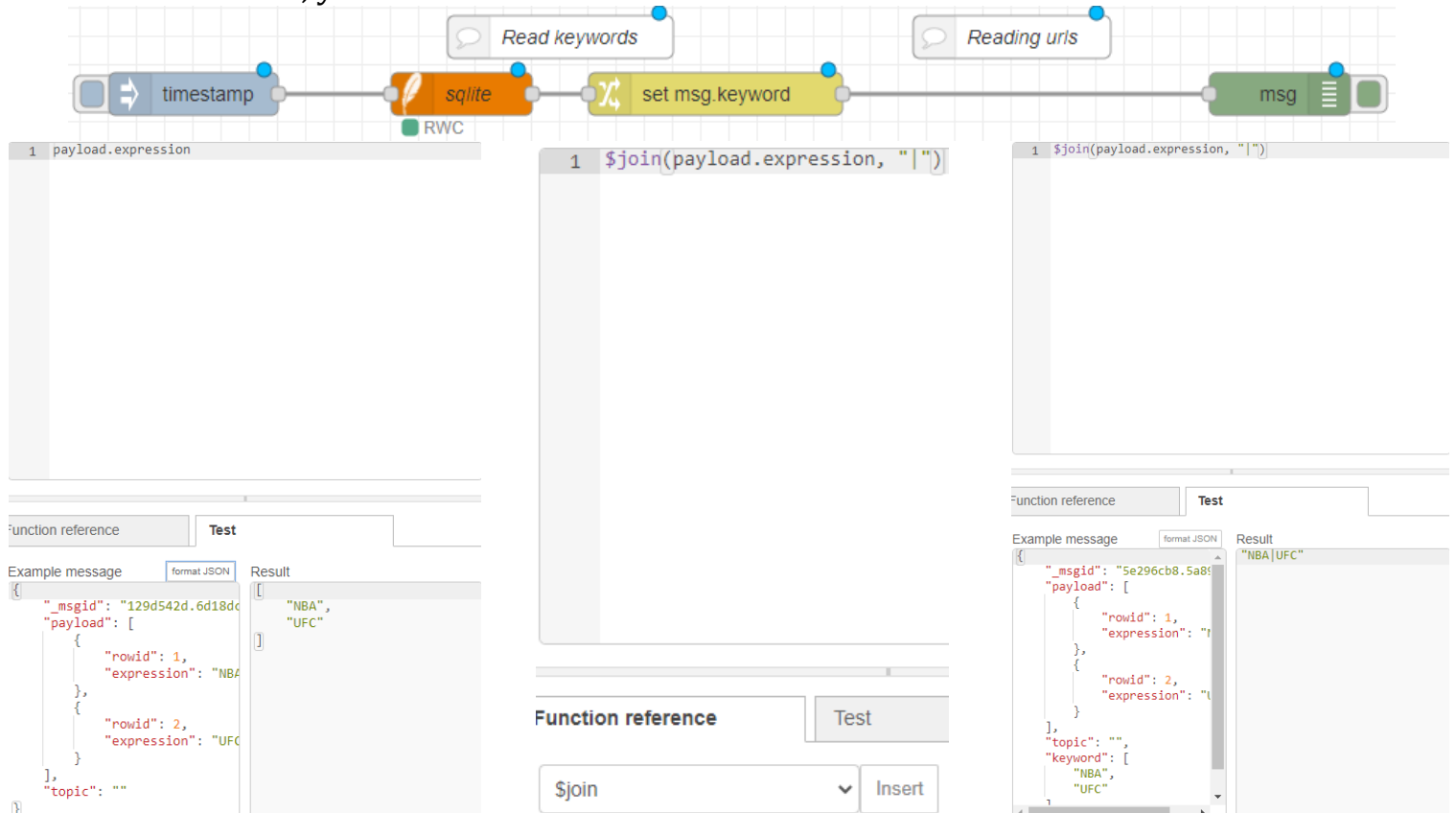


When you inject it, you should get your keywords as output.

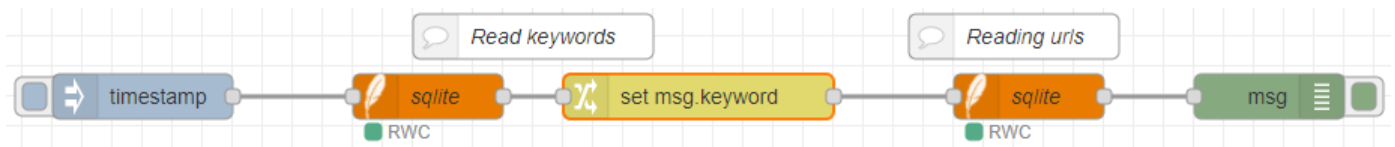


Next, add a change node, where we set the `msg.keyword` to a `Json` expression: `payload.expression`, and then join the outputs. Follow the steps below. You can also test the output to make sure it works. We are going to use the pipe(`"|"`) as an OR.

Also, you can add comments to better understand the flow.



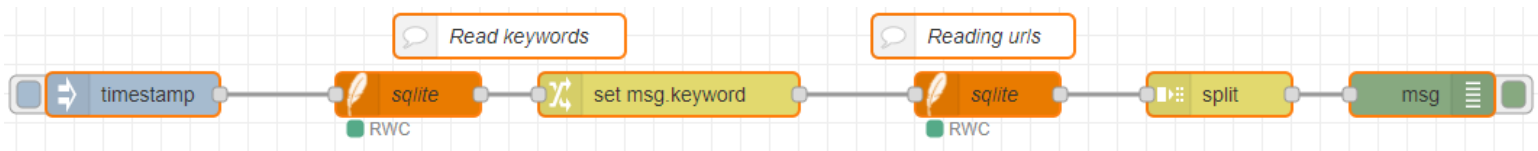
As shown above, the next step will be reading the feed urls. For that, we copy the fixed statement sqlite node from the feed flow.



You should get the keywords as an object property, displayed with pipes between them, and also the feed urls:

```
1/29/2021, 10:03:56 AM node: 8898fd85.b0b3
msg : Object
  object
    _msgid: "c883ab64.69e618"
    payload: array[3]
      0: object
      1: object
      2: object
        rowid: 4
        name: "Sports Keeda"
        url:
          "https://www.sportskeeda.com/feed"
        topic: ""
        keyword: "NBA|UFC"
```

To make it easier to forward these URLs, we can split the object into 3 individual objects. For this, we just use the split node.

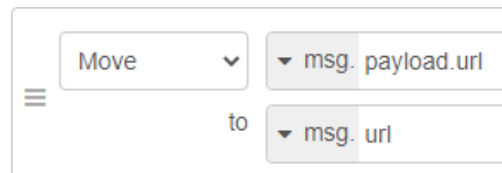


```
1/29/2021, 10:11:12 AM node: 8898fd85.b0b3
msg : Object
  { payload: object, topic: "",
    keyword: "NBA|UFC", parts: object,
    _msgid: "8fc958d6.9e8ec8" }

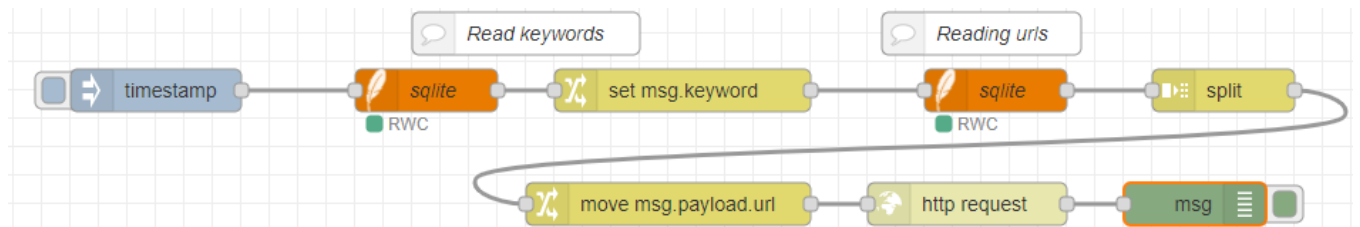
1/29/2021, 10:11:12 AM node: 8898fd85.b0b3
msg : Object
  { payload: object, topic: "",
    keyword: "NBA|UFC", parts: object,
    _msgid: "ff51ef93.c3297" }

1/29/2021, 10:11:12 AM node: 8898fd85.b0b3
msg : Object
  { payload: object, topic: "",
    keyword: "NBA|UFC", parts: object,
    _msgid: "dbfaefb.4083e1" }
```

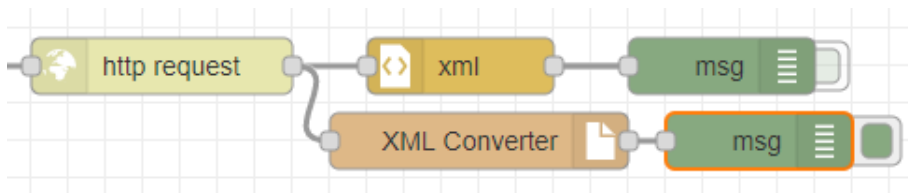
Add a change node after the split one, which MOVES the msg.payload.url to the msg.url



Add the http request and run the flow. You will have XML received.

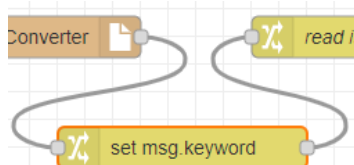
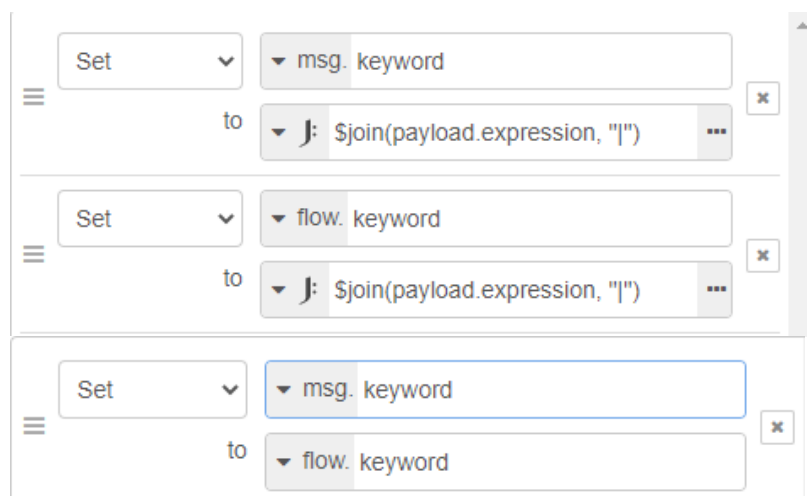
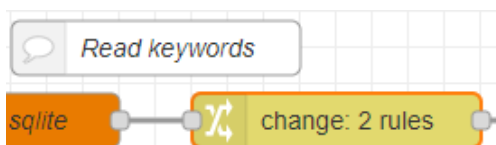


How do we process the XML? With the XML node. For me, the normal one did not work, so I had to install a new one, but this is the output you should be getting:



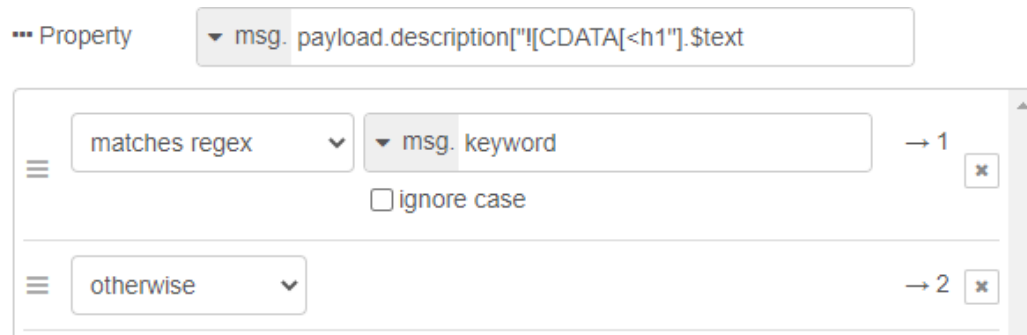
```
msg : Object
  ▼ object
  ▼ payload: object
    ▼ rss: object
      ▶ $namespaces: object
      version: "2.0"
      ▼ channel: object
        ▼ title: object
          $text: "FOX Sports Digital"
          ▶ link: object
```

When we do this, we might encounter some problems with the msg.keyword, so make the following modifications: where you first set the msg.keyword, also set the flow.keyword. And then, after the XML converter, add another set node that sets the msg.keyword to flow.keyword.

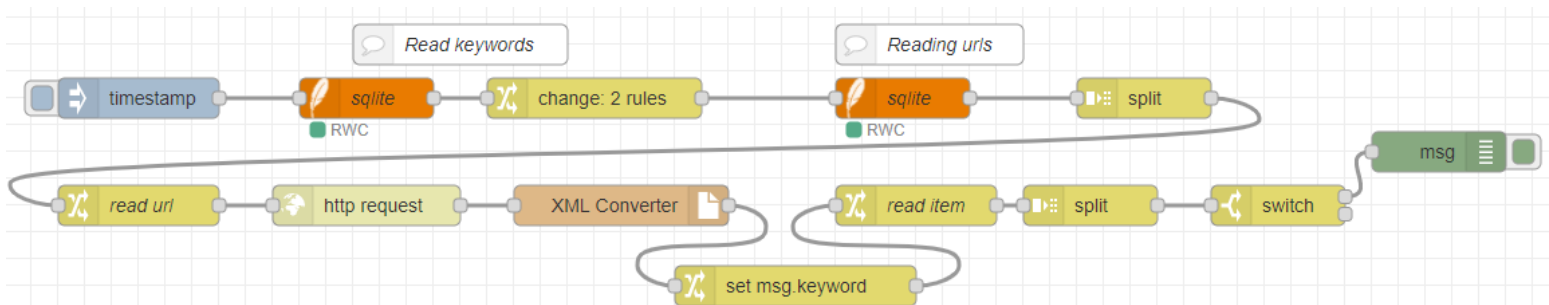


Next, we can follow the path for the area of interest to “msg.payload.rss.channel.item”. We move this to the msg.payload. Next, add a split and a switch node. The split node is left unchanged, and the switch node should have the following:

“payload.description["![CDATA[<h1"].\$text” – for copy pasta



To get the path to the text of the description, just copy the full path when you find an example. The new and scary flow:



When running, you should get some objects.



On the upper switch branch, add a change node, where we move the properties upwards.

Name: change

Rules:

- Move `msg.payload.description[\"!\[CDATA[<h1\"]]` to `msg.payload.description`
- Move `msg.payload.link.$text` to `msg.payload.link`
- Move `msg.payload.title.$text` to `msg.payload.title`

After it, add a function that removes the “ ‘ ” from the text, which can cause issues:

```
msg.payload.description = msg.payload.description.replace(/'/g,"")
msg.payload.link = msg.payload.link.replace(/'/g,"")
msg.payload.title = msg.payload.title.replace(/'/g,"")
```

```
return msg;
```

Copy a template DB insert node and edit it:

```
1 INSERT INTO match VALUES
2 (
3   '{{{payload.link}}}',
4   '{{{payload.title}}}',
5   '{{{payload.description}}}'
6 )
```

Add the DB node from the DB flow, which has an SQL query **Via** **msg.topic**.

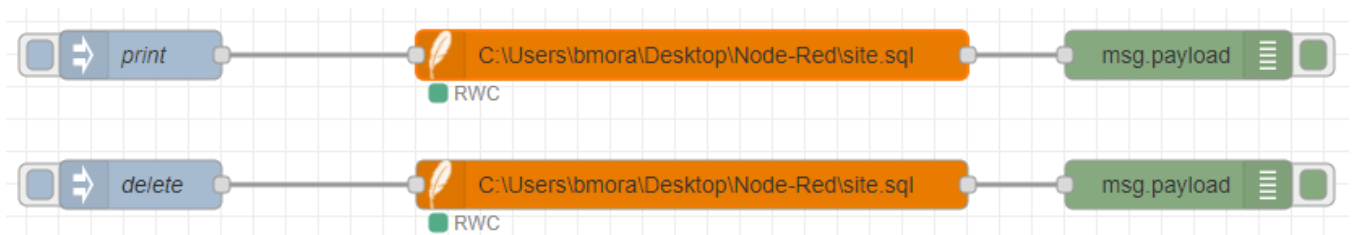
The output flow should look as follows:



Now at output we have what we need in the topic.

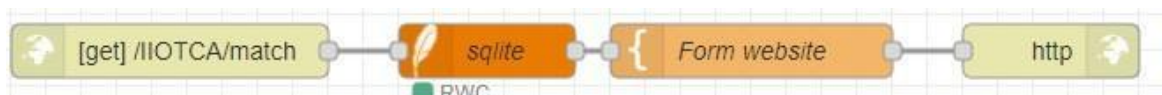
```
▼ object
  payload: array[0]
  keyword: "COVID|China|Trump|GME"
  ▶ parts: object
  _msgid: "a365d543.2bd098"
  ▼ topic: string
    INSERT INTO match VALUES
    (
    'https://www.buzzfeednews.com/art
    icle/janelytvynenko/holocaust-sur
    vivor-dies-covid-victims',
    '"One Of The Most Kind People You
    Would Ever Meet": She Escaped The
    Holocaust But Not The Pandemic',
    'Malvina Shabes escaped Poland du
    ring World War II and found refug
    e in Canada. A COVID-19 outbreak
    in a retirement home killed her.'
    )
```

To view what you have in the database, or delete the recordings, you can set up the two following flows.



In both of these, you only need to set up the sqlite node to either SELECT * FROM match or DELETE FROM match.

To output this match, just go to the Site flow, and copy the upper part of the feed flow, and replace feed with match wherever you can.



The last thing we need to edit here is the template node. We only need the title and summary here, and we can delete the form, so set it up accordingly.

```
<h1>Last news</h1>
<div class="row">
  <div class="col-md-12">
    <table class="table">
      <thead>
        <tr>
          <th scope="col">Title</th>
          <th scope="col">Summary</th>
          <th scope="col"></th>
        </tr>
      </thead>
      <tbody>
        {{#payload}}
        <tr class="table-primary">
          <td><a href="{{link}}">{{title}}</a></td>
          <td>{{description}}</td>
        </tr>
        {{/payload}}
      </tbody>
    </table>
  </div>
</div>
```

In the match page we can now see the news as a list.

Last news	
Title	Summary
On Day 1, Biden Directed The US To Rejoin The Paris Climate Accord	Under Trump's direction, the US became the only country to withdraw from the historic climate agreement.
Far-Right Extremists Around The World Are Drawing Inspiration From The Insurrection On Capitol Hill	On secret communication channels and open message boards, white supremacists are rejoicing at what President Donald Trump unleashed — and planning their next moves.
How One Company Made It Easier For An Autocrat To Crack Down, Then Lobbied Trump — And Won	Congo suffered under the rule of Joseph Kabila, but an audacious lobbying campaign helped him get what he wanted out of the Trump administration.
"One Of The Most Kind People You Would Ever Meet": She Escaped The Holocaust But Not The Pandemic	Malvina Shabes escaped Poland during World War II and found refuge in Canada. A COVID-19 outbreak in a retirement home killed her.
"I Forgot There Was A Pandemic": How It Feels Living In Places Where COVID Isn't A Thing	COVID infections are low to nonexistent in several countries, where life looks practically normal. Some people even occasionally forget there's a pandemic going on.
Inside A Xinjiang Detention Camp	In a lush countryside idyll known for its horse farms and fields of yellow flowers, China built a system of total control.
Heres How Newspapers Across The US And Around The World Covered Bidens Win (And Trumps Loss)	Biden defeating Trump was front-page news from Iowa to Alaska to Japan.
It's A US Territory Where The Coronavirus Never Arrived — But Some Residents Can't Get Home	The remote US territory of American Samoa has not recorded a single case of COVID-19 after shutting itself off to the world in March. Even some locals cant get home.

Congrats, you made a full stack app using node-red.



Please don't be a cheetah.

