

DATA MANAGEMENT WITH RULES ENGINE

Storing telemetry data from IoT devices in a database for later retrieval is useful for a variety of reasons. It can help you or your audience understand changes in environmental readings over time, serve as input to a machine learning model, and more.

In this tutorial, we'll save the data in a database. Many databases are supported by the Rules Engine but this tutorial will use:

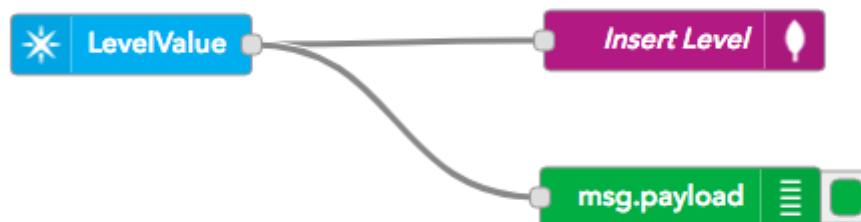
- MongoDB
- Firebase

Using the Rules Engine makes it easy to manipulate the data published from the device into the exact format you want to store in your database.

We'll continue to use the device level sensor and device firmware from the first tutorial, [Real-time alerting](#).

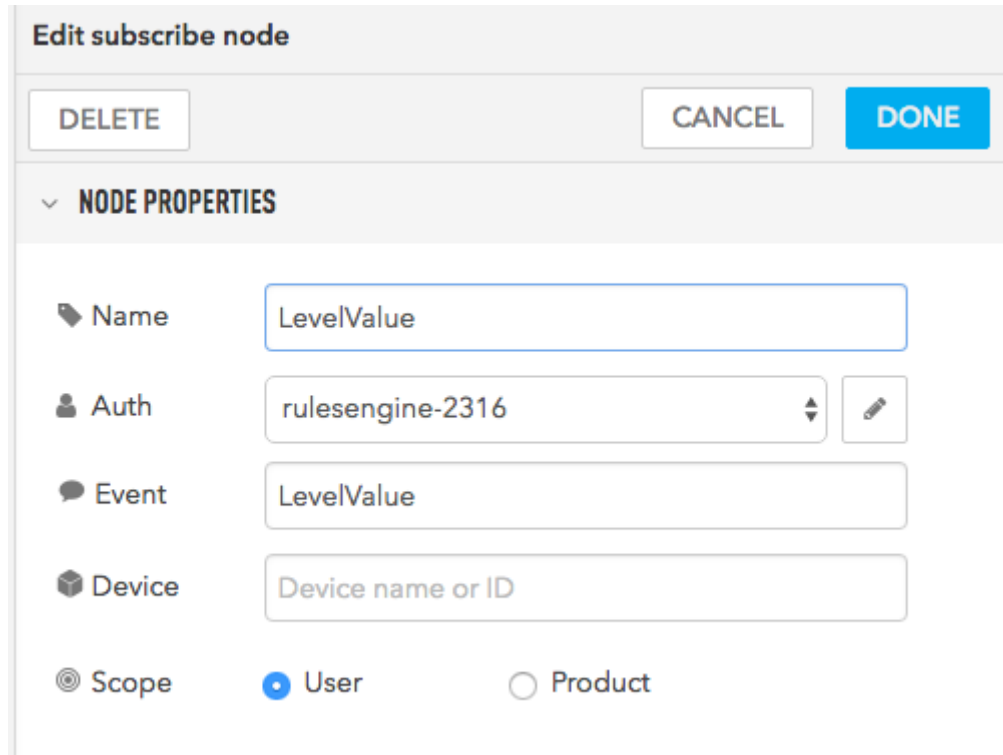
Creating the Flow

We'll be starting with a new flow because there are number of changes from the last one.



- From the **Particle** group, drag **subscribe** node to your flow.
- Double click to edit it.
- Set **Name** to **LevelValue** (can be anything).

- Set **Auth** to the authentication we created in the real-time alerting tutorial.
- Set **Event** to **LevelValue**.
- Leave the **Device** field empty
- Leave the **Scope** at **User**.
- Click **Done**.



Edit subscribe node

DELETE CANCEL DONE

▼ **NODE PROPERTIES**

Name: LevelValue

Auth: rulesengine-2316

Event: LevelValue

Device: Device name or ID

Scope: ☒ User ☐ Product

- From the **output** group drag a **debug** node to the flow as well. Leave the default as outputting **msg.payload**.



- You can use this as a basis for any of the database tutorials.

MongoDB Tutorial

Getting a database

This example uses MongoDB. If you don't already have a MongoDB instance you can connect to, or have one installed on your computer, you can also use a hosted solution like [MongoDB atlas](#) that takes care of all of the hard work.

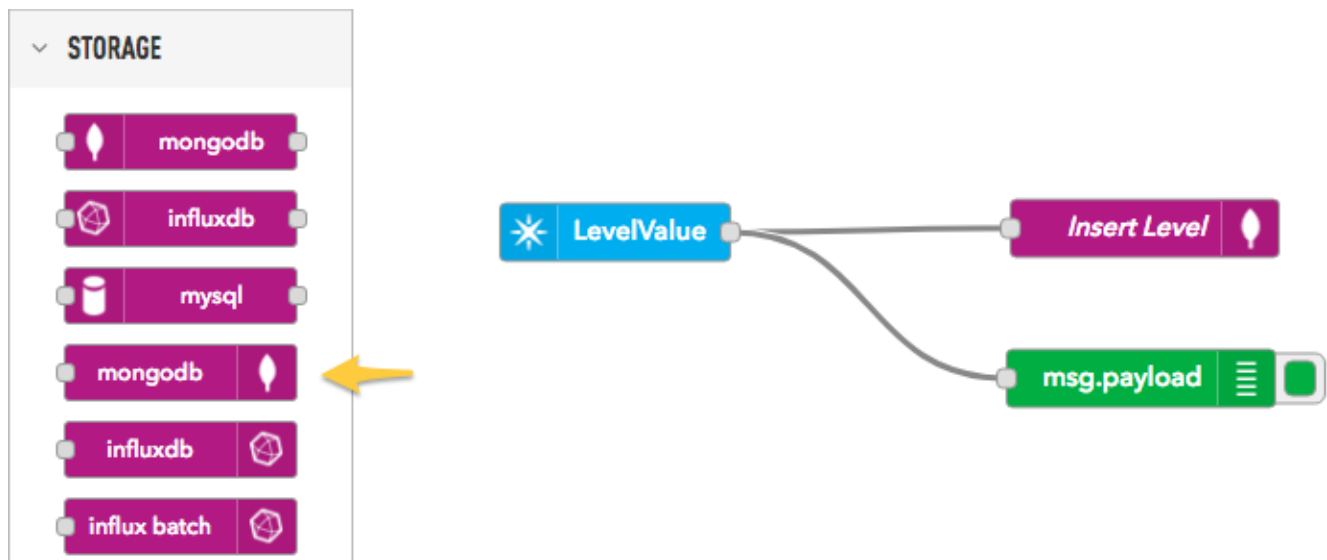
Creating the MongoDB flow

Drag the Copy Rules button into the Rules Engine window to create the flow automatically, or you can create the flow from scratch with the steps below.

Copy Rules



- From the **Storage** section drag the **mongodb** (out) node to your flow.



- Double click on the **mongodb** node to edit the configuration.
- Click on the pencil icon to edit the server configuration.


Edit mongodb out node

DELETE


CANCEL


DONE


▼ NODE PROPERTIES

 Server

Add new mongodb...




 Collection

 Operation

save

☐

 Only store msg.payload object


 Name


Name


Tip: If no collection is set, ensure `msg.collection` will contain the collection name


- In **Host** enter the hostname of your mongodb server. You cannot use a locally hosted server (127.0.0.1) since the Rules Engine is hosted remotely.
- In **Database** enter the database name.
- In **Username** enter your database username. Since you can't use a locally hosted MongoDB, you will almost certainly need a username and password.
- In **Password** enter the password.
- In **Name** you can enter whatever name you'd like.


Edit mongodb out node > **Add new mongodb config node**

 Host Port

 Database

 Username

 Password

 Name

- After configuring the Server, set the remaining items.
- In **Collection** specify the collection that you will be storing into.
- In **Operation** change to **Insert** because you want to add a new item to the collection for each publish.
- Leave the **Only store msg.payload object** checkbox unchecked.
- Set **Name** to whatever you'd like.

Edit mongodb out node

DELETECANCELDONE

▼ NODE PROPERTIES

Server

My Mongo DB

Collection

level

Operation

insert

☐ Only store msg.payload object

Name

Insert Level

- If you haven't connected all of the nodes up yet, connect them up now.
- Deploy your flow.
- In the Debug panel on the right, you can watch events as they arrive:

8/1/2018, 2:42:25 PM node: ac961fdd.05169

msg.payload : string(8)

"0.769231"

8/1/2018, 2:43:25 PM node: ac961fdd.05169

msg.payload : string(8)

"0.770452"

8/1/2018, 2:44:25 PM node: ac961fdd.05169

msg.payload : string(8)

"1.323565"

8/1/2018, 2:45:25 PM node: ac961fdd.05169

msg.payload : string(8)

"1.322344"

- If you view the database (in this case, with Robo3T), you can see all of the fields that were automatically added to the database.

```
db.getCollection('level').find({})
```

level 0.005 sec.

Key	Value	Type
(1) Objectid("5b61fc6d7ef7ad00152dec77")	{ 6 fields }	Object
_id	Objectid("5b61fc6d7ef7ad00152dec77")	Objectid
event	LevelValue	String
payload	0.196581	String
published_at	2018-08-01T18:31:09.216Z	String
device_id		String
_msgid	80f53481.ffafa8	String

- Or, more usefully, in table view:

```
db.getCollection('level').find({})
```

level 0.003 sec.

	_id	event	payload	published_at	device_id	_msgid
1	Objectid(...)	LevelValue	0.196581	2018-08-01T18:31:09.216Z	1e00...	80f5348...
2	Objectid(...)	LevelValue	0.196581	2018-08-01T18:32:25.690Z	1e00...	3b5bb7e...
3	Objectid(...)	LevelValue	0.769231	2018-08-01T18:33:25.692Z	1e00...	a4df184...
4	Objectid(...)	LevelValue	0.769231	2018-08-01T18:34:25.690Z	1e00...	70982e6...
5	Objectid(...)	LevelValue	0.768010	2018-08-01T18:35:25.689Z	1e00...	bb88127...
6	Objectid(...)	LevelValue	0.769231	2018-08-01T18:36:25.689Z	1e00...	71444dc...
7	Objectid(...)	LevelValue	0.769231	2018-08-01T18:37:25.688Z	1e00...	6691b20...
8	Objectid(...)	LevelValue	0.769231	2018-08-01T18:38:25.690Z	1e00...	88909fd...
9	Objectid(...)	LevelValue	0.769231	2018-08-01T18:39:25.687Z	1e00...	997697b...
10	Objectid(...)	LevelValue	0.770452	2018-08-01T18:40:25.686Z	1e00...	7b4b3d1...
11	Objectid(...)	LevelValue	0.769231	2018-08-01T18:41:25.686Z	1e00...	c12582d...
12	Objectid(...)	LevelValue	0.769231	2018-08-01T18:42:25.685Z	1e00...	526edfe...
13	Objectid(...)	LevelValue	0.770452	2018-08-01T18:43:25.684Z	1e00...	908061ff...
14	Objectid(...)	LevelValue	1.323565	2018-08-01T18:44:25.684Z	1e00...	6bed87e...
15	Objectid(...)	LevelValue	1.322344	2018-08-01T18:45:25.686Z	1e00...	6df1e1ad...
16	Objectid(...)	LevelValue	1.323565	2018-08-01T18:46:25.700Z	1e00...	76e031f...
17	Objectid(...)	LevelValue	1.323565	2018-08-01T18:47:25.689Z	1e00...	acb839f...
18	Objectid(...)	LevelValue	1.323565	2018-08-01T18:48:25.694Z	1e00...	f8380d6...


Google Firebase

Another place you can easily save data is in Google Firebase.

Setting up Firebase




- Go to the [Firebase Console](#)
- Add a project. I named mine **rules-engine-1**


Add a project


 You're 3 projects away from the project limit.


Project name


rules-engine-1


 +  + 

Tip: Projects span apps across platforms 

Project ID 

rules-engine-1 

Analytics and billing region 



☒ Use the default settings for sharing Google Analytics for Firebase data

☒ Share your Analytics data with Google to improve Google Products and Services

☒ Share your Analytics data with Google to enable technical support

☒ Share your Analytics data with Google to enable Benchmarking

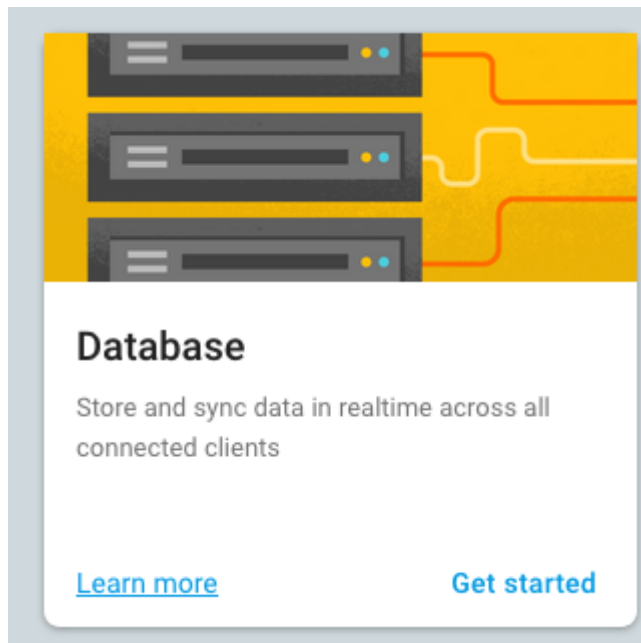
☒ Share your Analytics data with Google Account Specialists

☒ I accept the [controller-controller terms](#). This is required when sharing Analytics data to improve Google Products and Services. [Learn more](#)

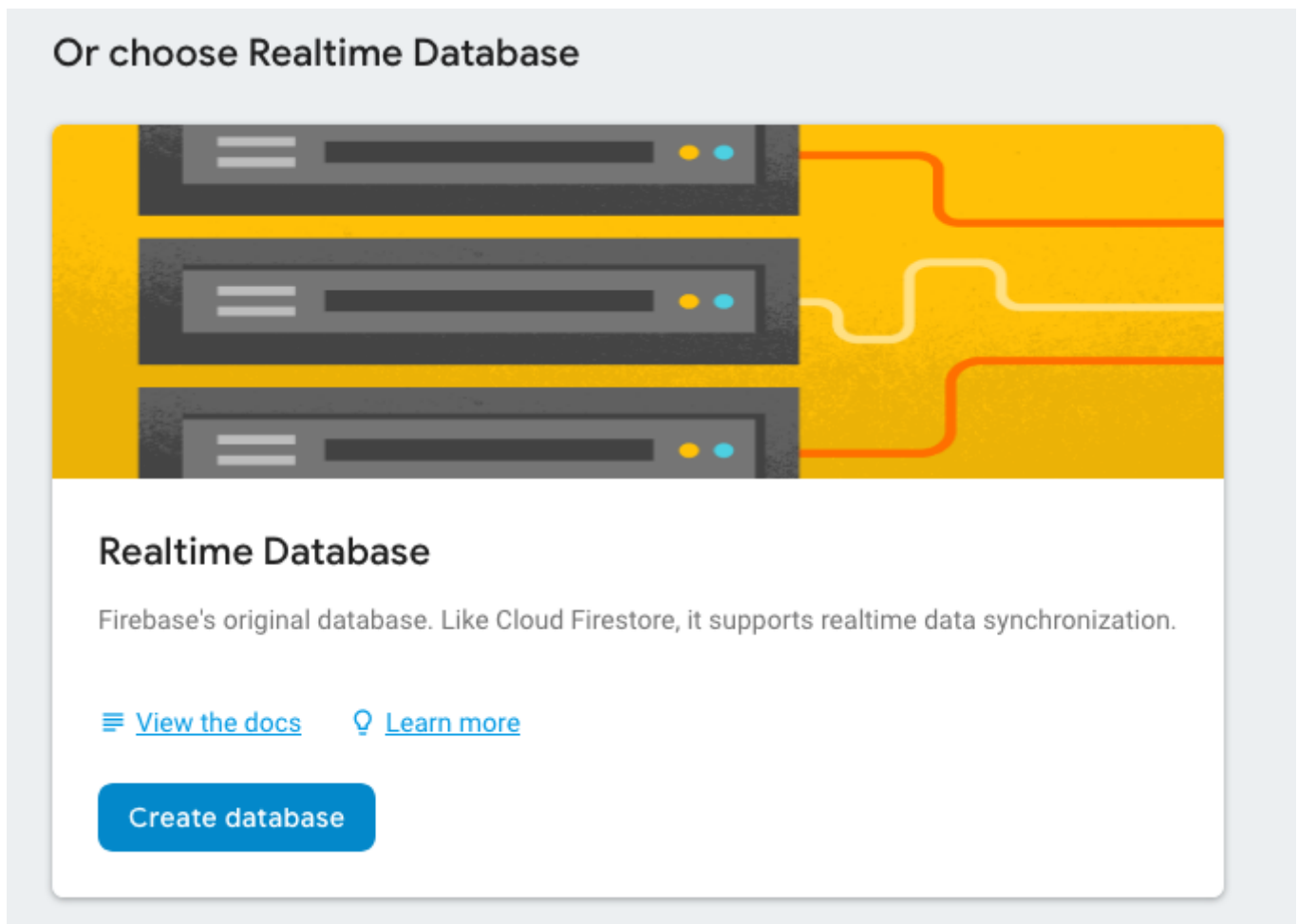
Cancel

Create project

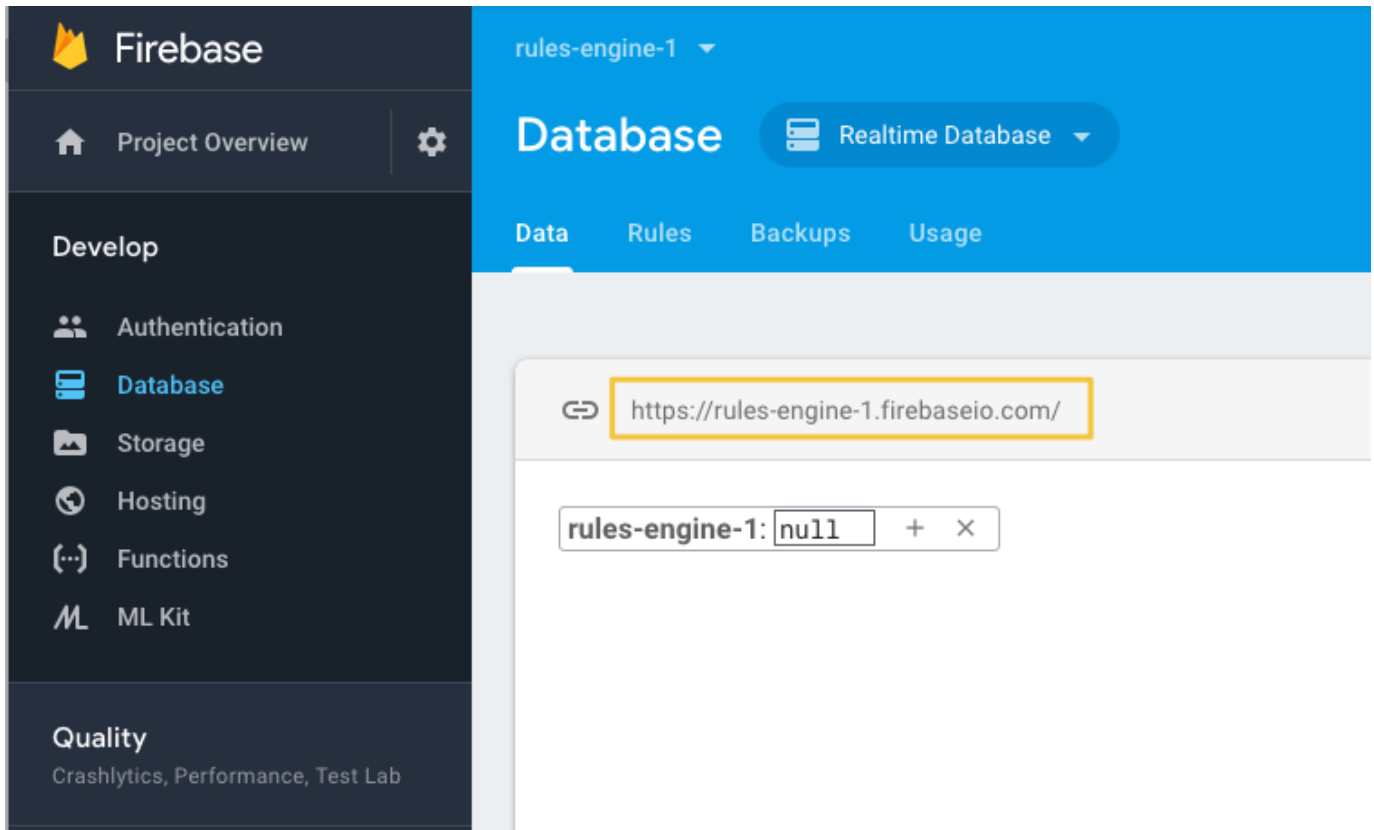
- Click **Get Started** in the **Database** section.



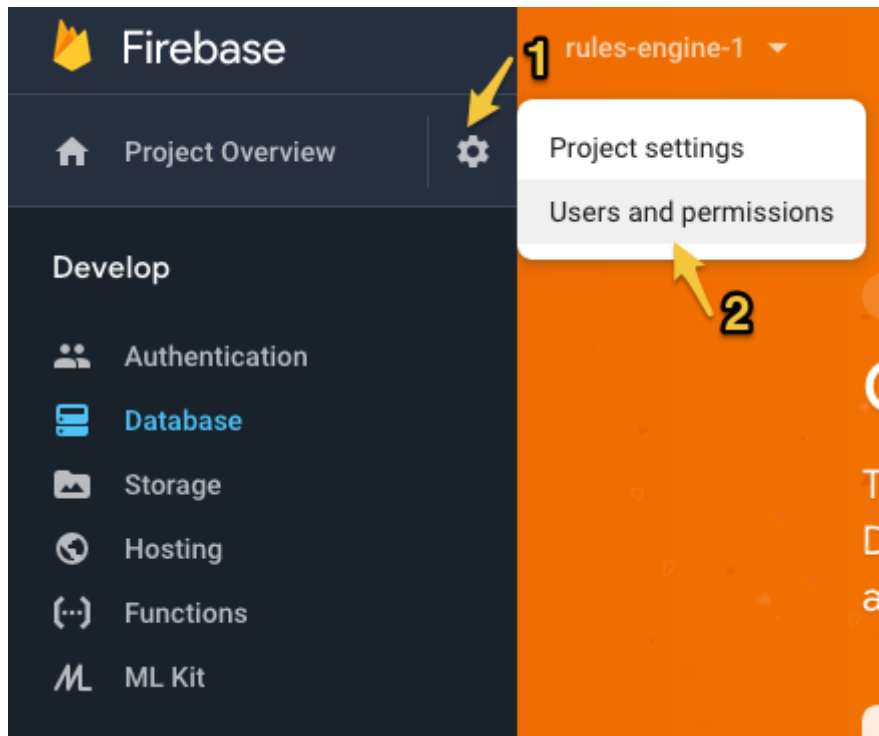
- We'll be using the original **Realtime Database** in this example.



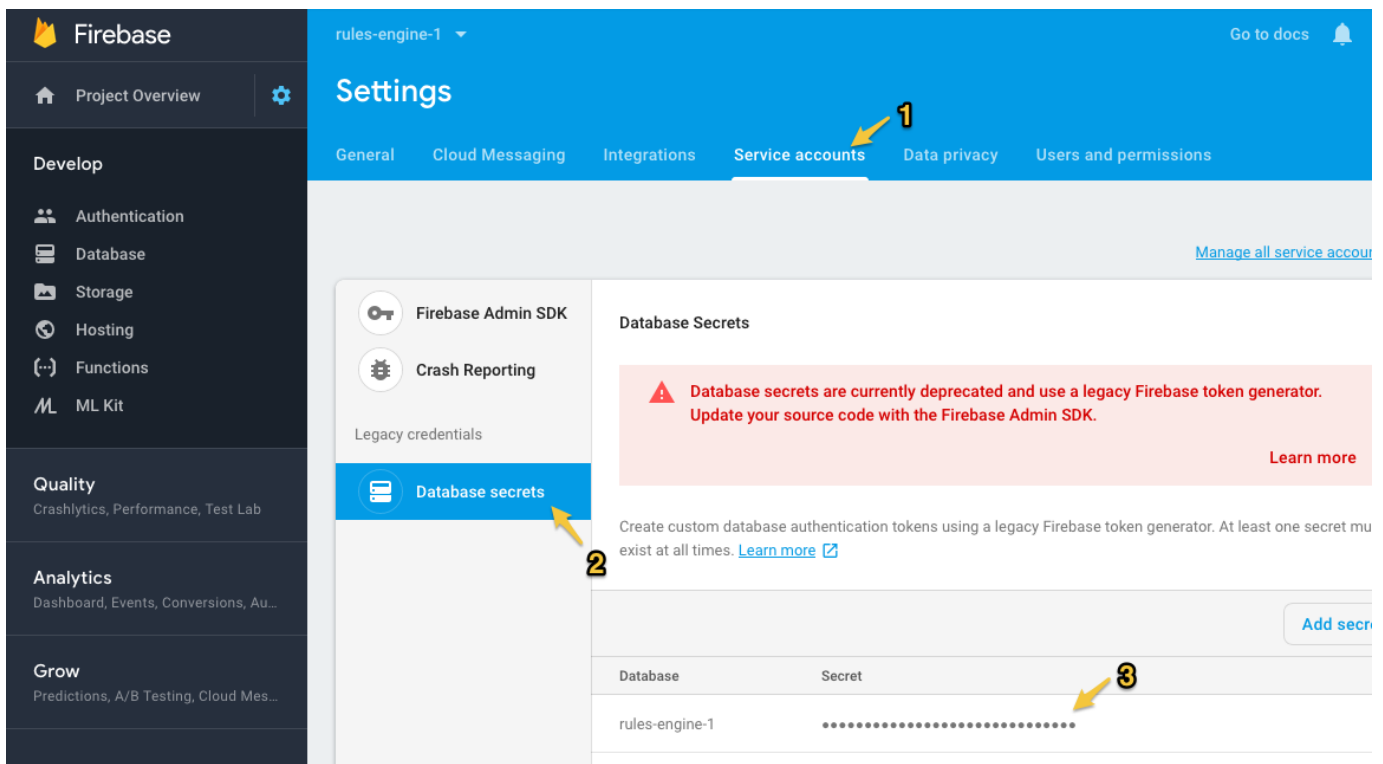
- You should use the locked settings. We're using an authentication token to write to the database so the locked settings will work perfectly.
- In the main Database window, note your Firebase URL. In this example, it's `https://rules-engine-1.firebaseio.com`.



- In the main Firebase console window, select the gear icon next to **Project Overview** (1) in the upper left.
- Then select **Users and permissions** (2).

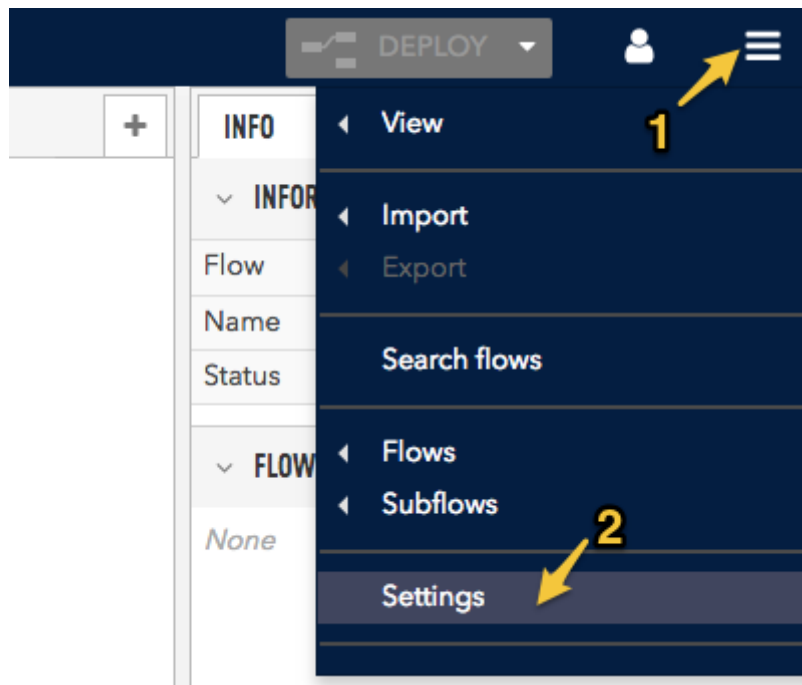


- Select **Service Accounts** (1).
- Select **Database secrets** (2).
- Copy the secret key (3). Even though the Firebase secrets are technically deprecated, they're the easiest way to authenticate the Firebase node.

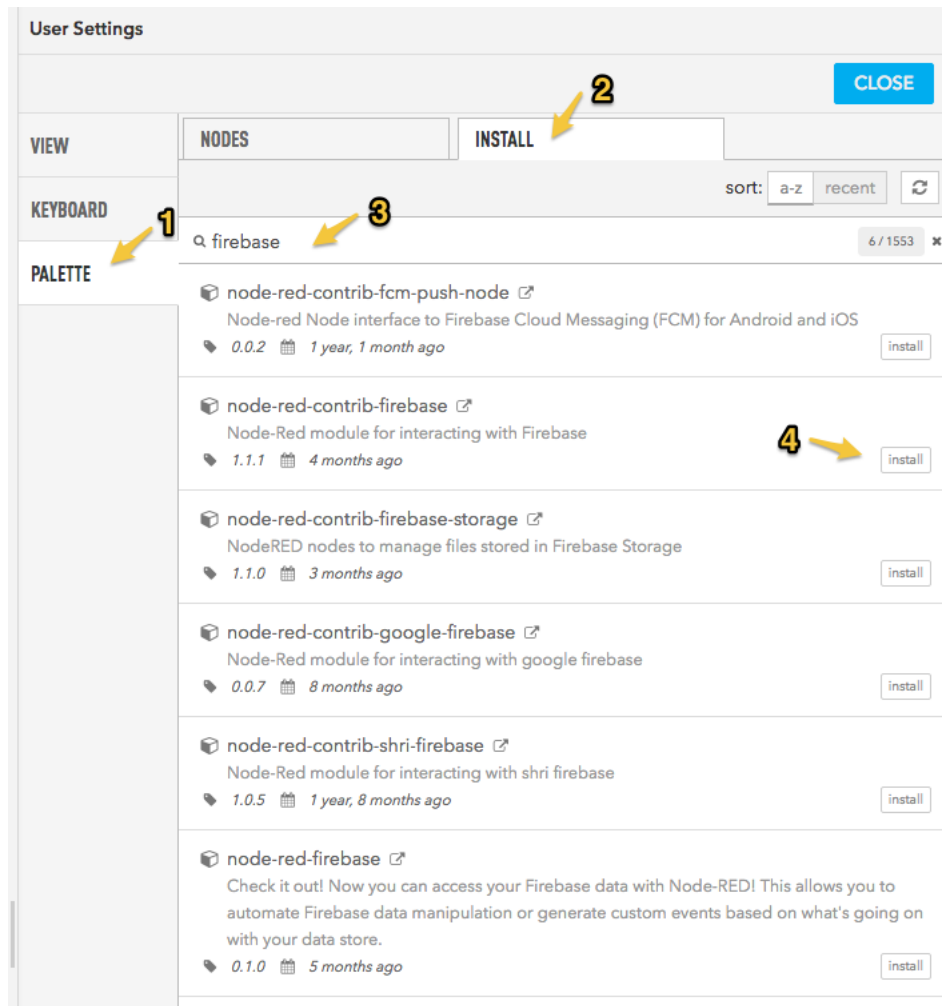


Adding Firebase to the Rules Engine Palette

- In the Rules Engine, click the "hamburger icon" in the upper right of the rules engine window (1) then **Settings** (2).

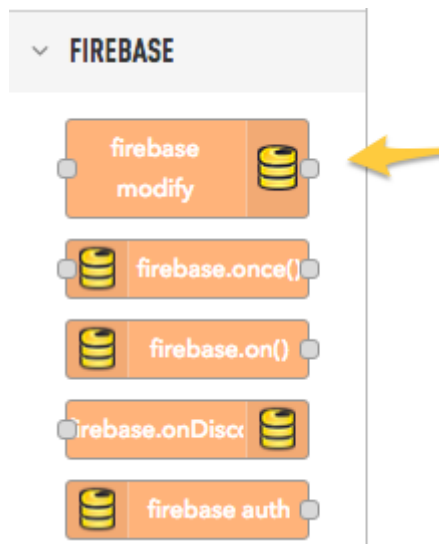


- Click **Palette** (1).
- Click **Install** (2).
- Type in the search field **firebase** (3).
- Click **Install** (4) for **node-red-contrib-firebase**.

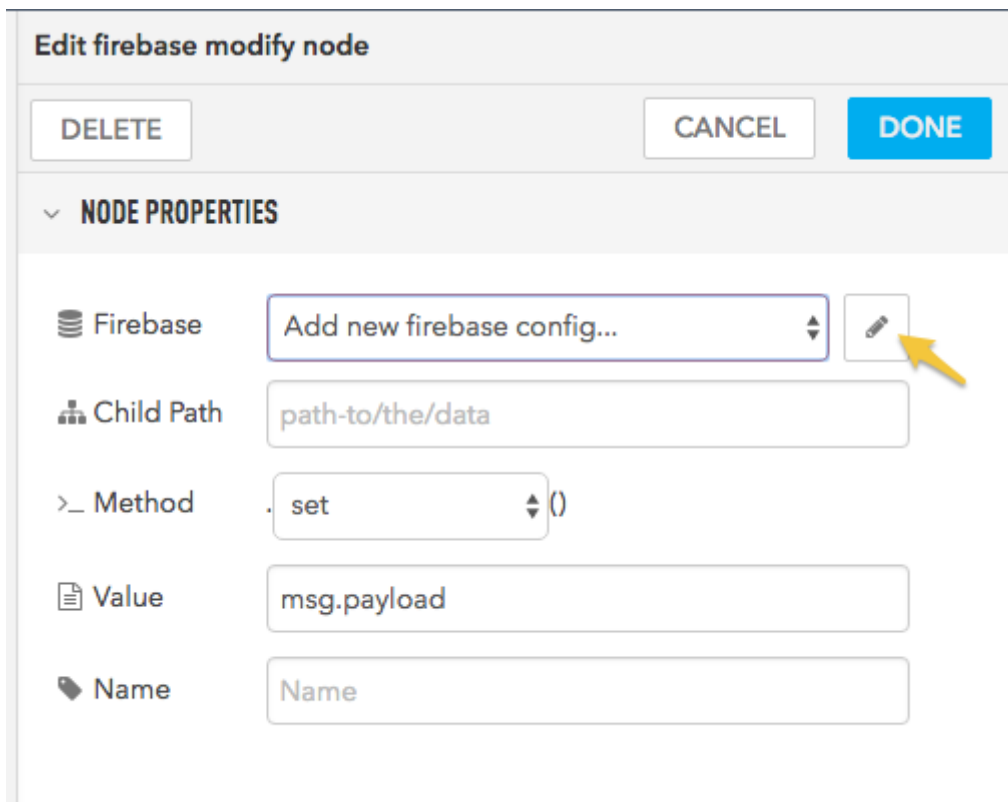


Setting up the firebase flow

- From the **Firebase** section, drag the **firebase modify** (out) node to your flow.





- Double click the **firebase modify** node to edit it.
- Click the pencil icon to **Add new firebase config...**





Edit firebase modify node


DELETE CANCEL DONE


▼ **NODE PROPERTIES**

 **Firebase** Add new firebase config... 

 **Child Path** path-to/the/data

 **Method** set ()


 **Value** msg.payload


 **Name** Name


- Edit the **Firebase** item with your Firebase URL.
- Set the **Auth Type** to **JSON Web Token**.
- Copy and paste in your **Database Secret**.
- Click **Update**.

Edit firebase modify node > **Edit firebase config node**

DELETE CANCEL UPDATE

 **Firebase** https:// rules-engine-1 .firebaseio.com/

 **Auth Type** JSON Web Token


 **Secret**


- Continue editing in the **Edit firebase modify node**.
- Set the **Child Path** to the key to hold your values. I used **levels**.
- Set the **Method** to **Push**. This creates a new entry for each event.
- Set the **Value** to **msg.payload**.
- Set the **Name** to whatever you want. I used **Write to Firebase**.
- Click **Done**.


Edit firebase modify node


DELETE CANCEL DONE


▼ **NODE PROPERTIES**

 **Firebase** https://rules-engine-1.firebaseio.com/ - .

 **Child Path** levels

 **Method** push ()

 **Value** msg.payload

 **Name** Write to Firebase

Setting up the flow

Drag the Copy Rules button into the Rules Engine window to create the flow automatically, or you can create the flow from scratch with the steps below.

Copy Rules ►

This is the flow we'll be creating:



- The flow begins with **subscribe** node from the **Particle** group. Set the settings as for other subscribe nodes in this tutorial.

Edit subscribe node


DELETE

CANCEL


DONE

▼


NODE PROPERTIES


 Name


LevelValue

 Auth


rulesengine-2316






 Event

LevelValue

 Device

Device name or ID

 Scope

☒ User

☐ Product

- From the **Function** group, drag a **function** into the flow. (Note: this is the function function, not the Particle function!)
- Set the **Name** to **Format Database Data** (or whatever you want)
- Set the function to:

```
msg.payload =  
{'level':parseFloat(msg.payload),'published_at':msg.published_at};  
  
return msg;
```

Edit function node


DELETE


CANCEL

DONE

▼ NODE PROPERTIES

 Name



 Function

```
1 msg.payload = {'level':parseFloat(msg.payload), 'published_at':new Date().toISOString()}
2
3 return msg;
```

This determines what data will be uploaded to Firebase. In this case, a floating point value containing the level and a timestamp from the event.

- From the **Output** section of the palette, drag a **debug** into your flow. This makes debugging easier.
- Finally, connect all of the nodes as shown above.
- Deploy your flow and hopefully everything will work.
- Viewing the **Debug** tab on the right side of the window should show data as it's published.



- And viewing the database in the Firebase console should show the data.

rules-engine-1

levels

-LlrVPAsrRXy9SsxEZJC

level: 0.096459

published_at: "2018-08-01T22:00:25.592Z"

-LlrVcplsaPpjNZi6lca

level: 0.09768

published_at: "2018-08-01T22:01:25.589Z"

-LlrVrTnQUvUfMxQQi40

level: 0.09768

published_at: "2018-08-01T22:02:25.591Z"