Bluemix IoT Watson Lab

Objective :

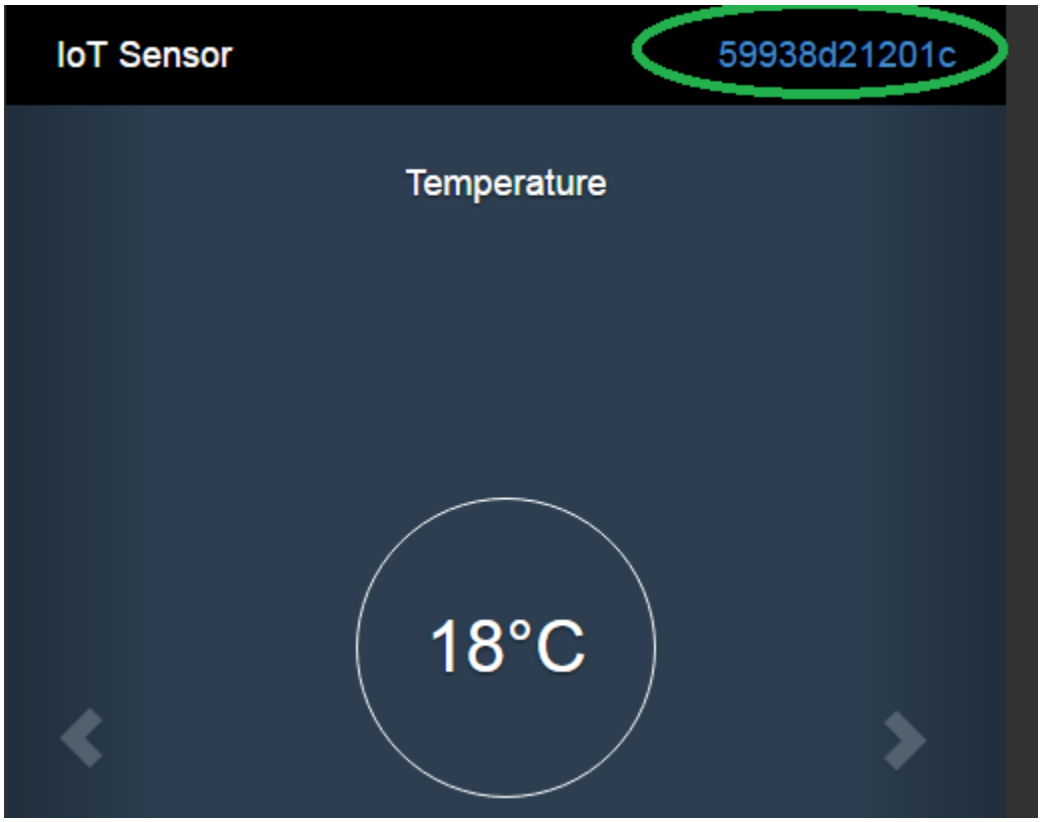
- Moving temperature data to cloud storage Cloudant , explore using cloudant queries , map-reduce , search indexes
- Explore node-red to handle IoT data , using nodes from IoT platform , function nodes,switch nodes and debug nodes
- Create data warehouse and replicate the data from no-sql (cloudant) to sql based (dashDB) database. Analyze using SQL queries and R Script
- Exercises included for self exploration

Duration : 2.5 hours

Rajesh K Jeyapaul
Cloud Solution Architect, IBM
July 2016

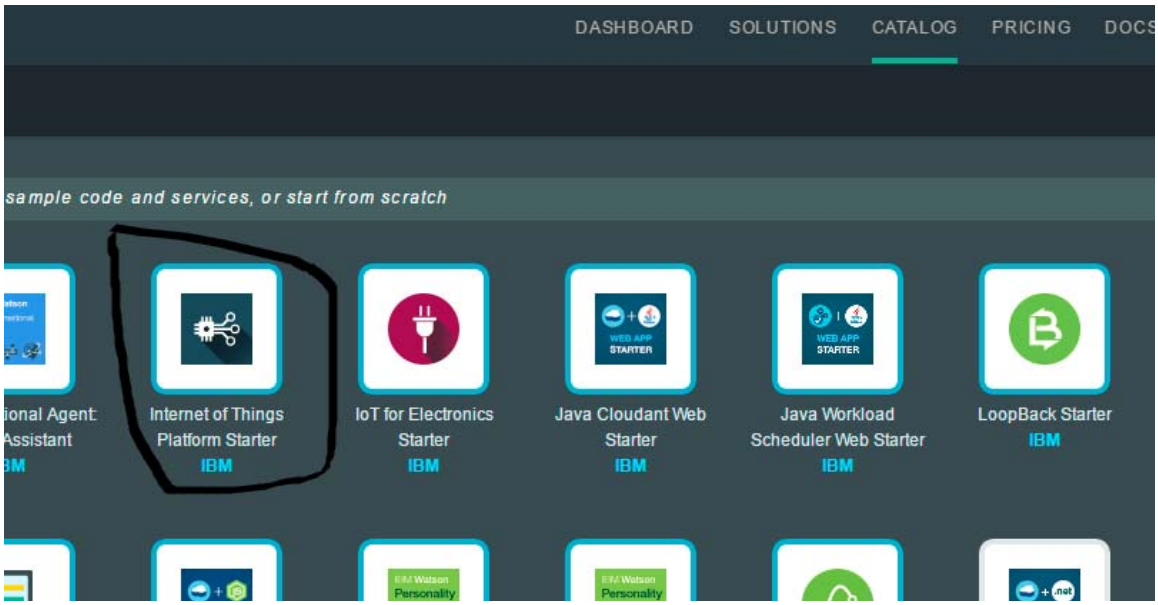
Lab 1: Create Bluemix Watson IoT Platform and move simulator data to cloud storage

➔ connect to the IoT simulator and note down the device id
- https://quickstart.internetofthings.ibmcloud.com/iotsensor/

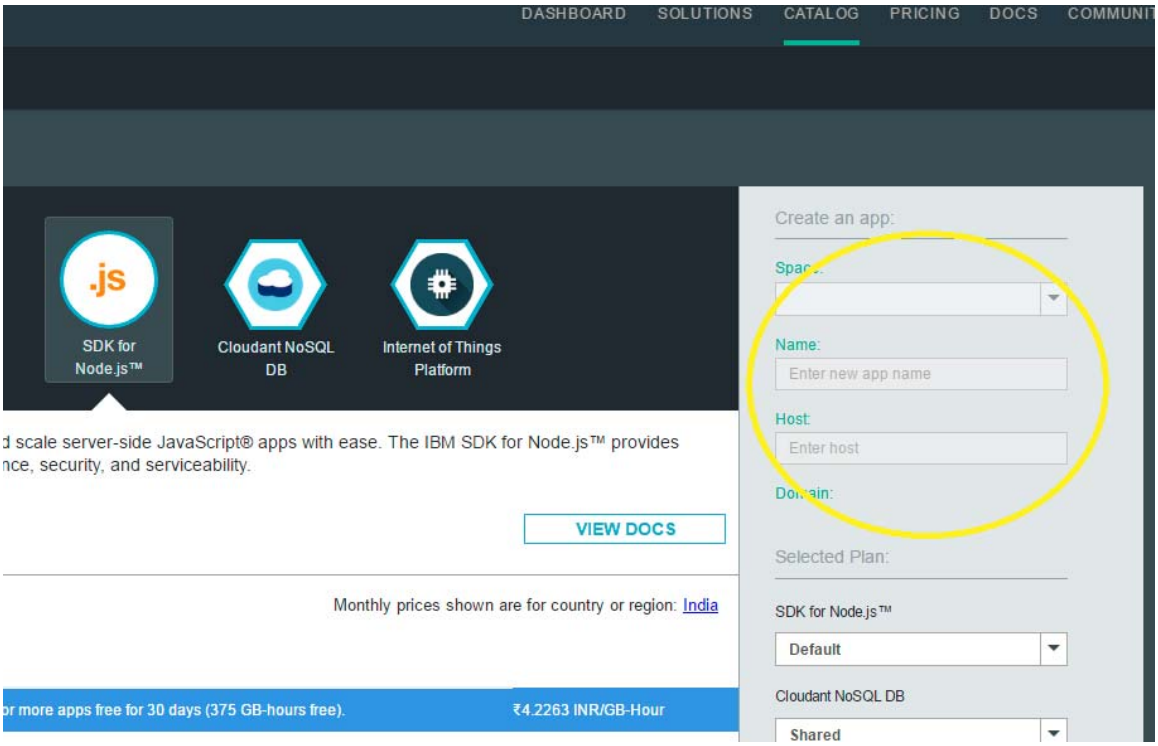


➔ **Deploy Internet of Things starter Boilerplate**

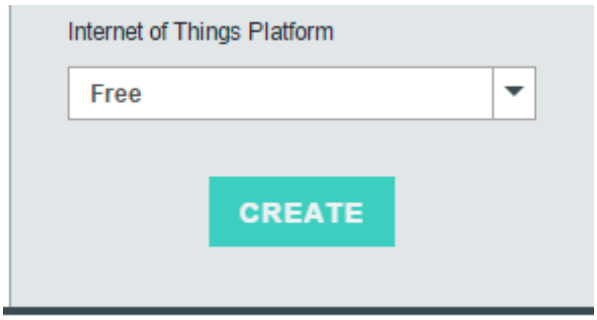
Go to Bluemix catalog page and select “Internet of Things Platform Starter” as shown below:



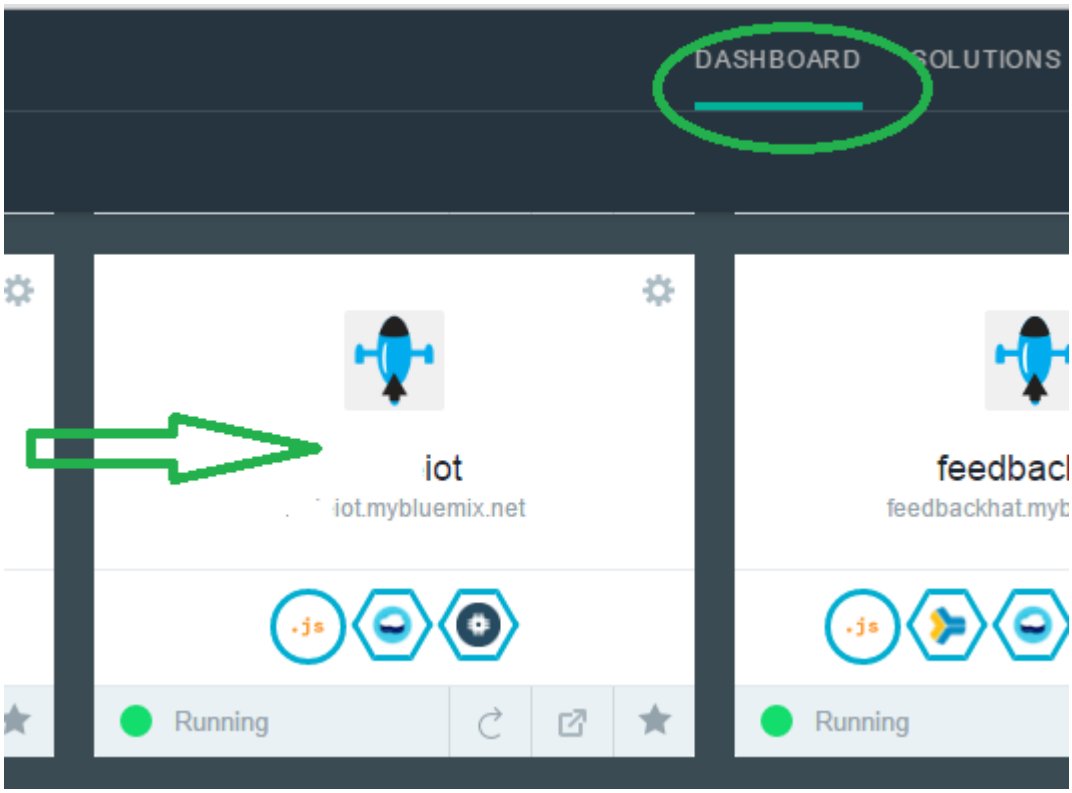
Provide an unique name for your application as shown:



Proceed to create



Go back to Dashboard and select the application that got deployed just now as shown



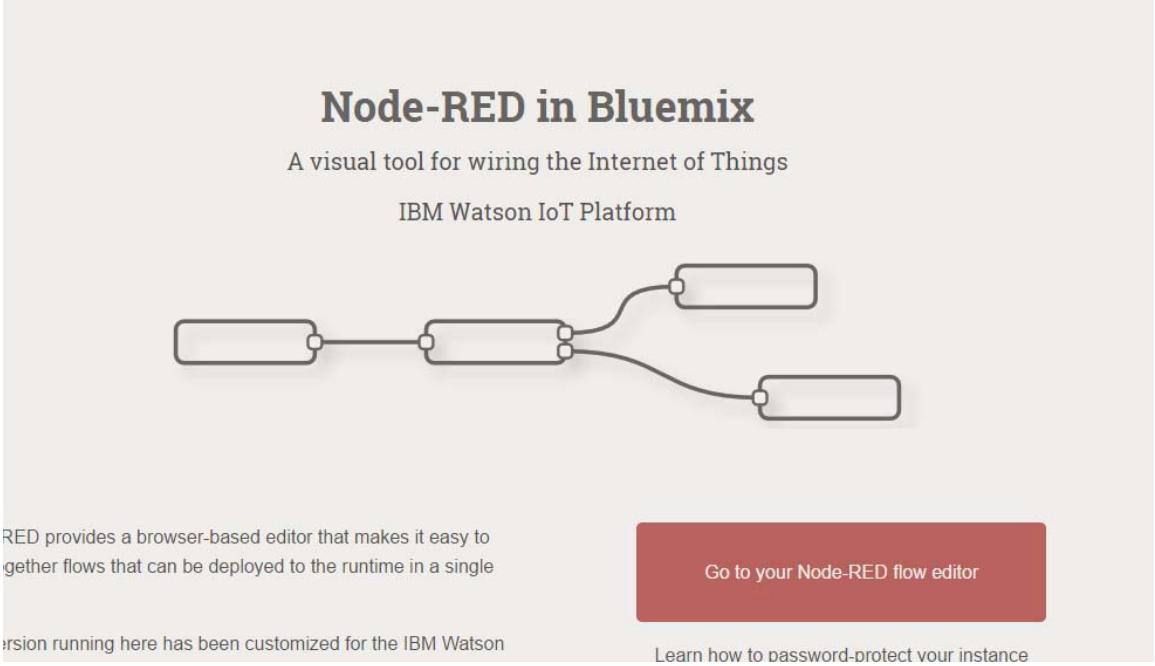
Run the application using the route provided, in this case:

iot.mybluemix.net

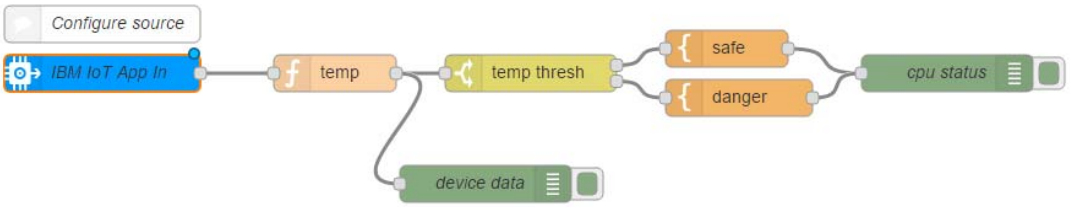
Note; This is the application got created with nodejs server , cloudant database and the IoT platform

On running the application , the nodejs server provides a node-red editor to explore the IoT platform and the data

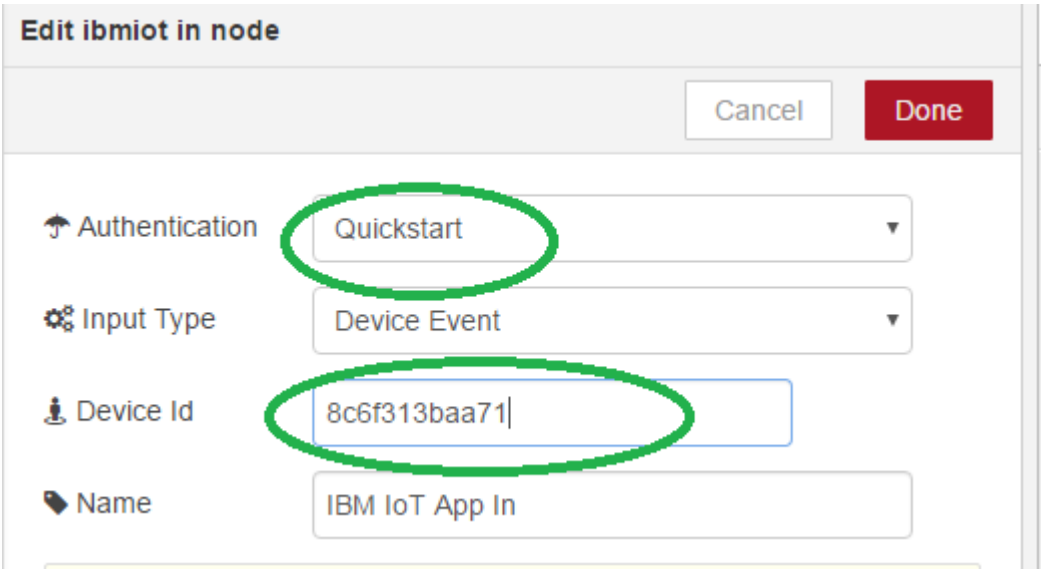
Proceed with “Go to your Node-RED flow editor”



We will use the default flow available to move the simulator data to the cloudant database for further exploration



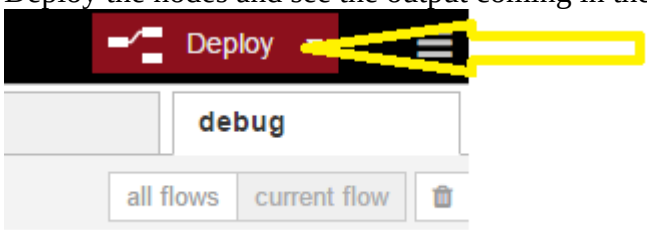
Double Click on the “IBM IoT App In” to provide the device ID

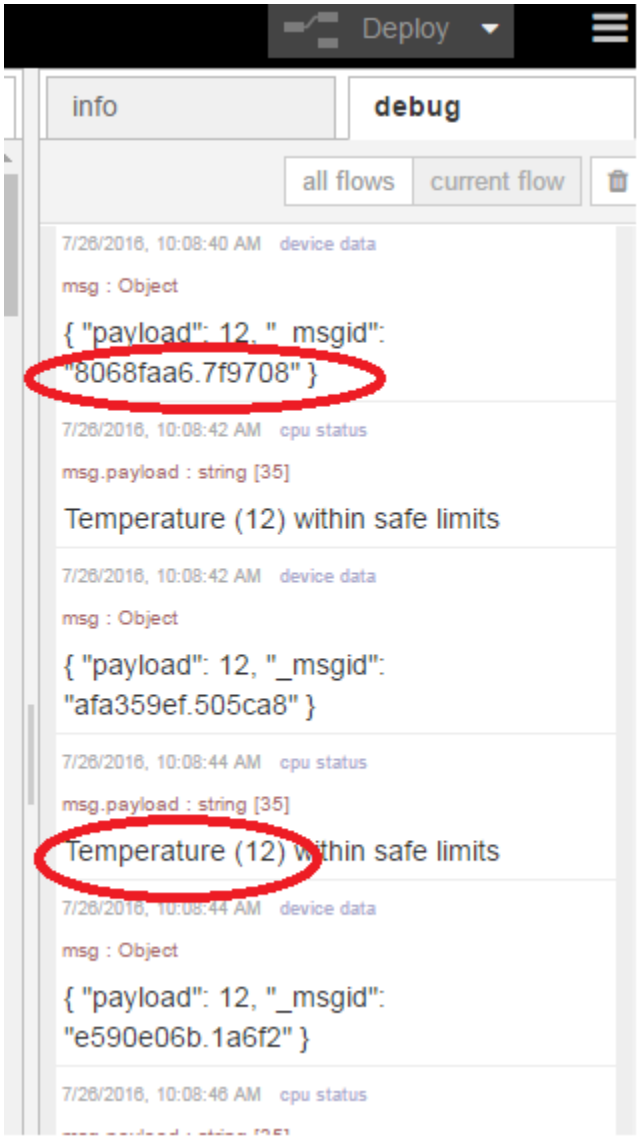


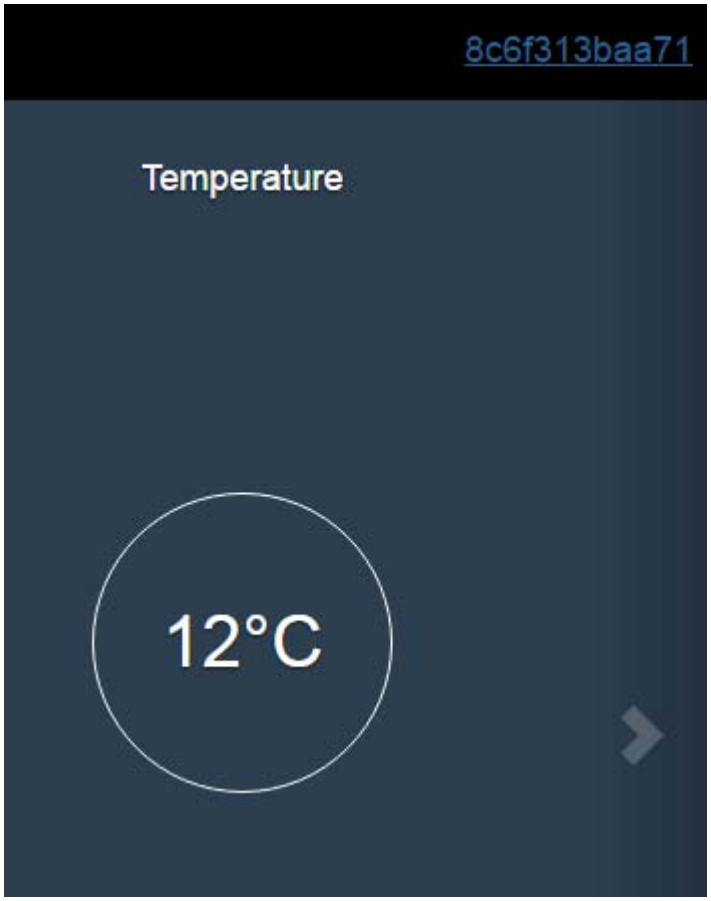
Ensure that the authentication be “Quickstart”

Note: Quickstart does not need any device registration and it publishes the data without any auth tokens

Deploy the nodes and see the output coming in the debug console:




Deploy
info
debug
all flows
current flow
7/26/2016, 10:08:40 AM device data
msg : Object
{ "payload": 12, "_msgid": "8068faa6.7f9708" }
7/26/2016, 10:08:42 AM cpu status
msg.payload : string [35]
Temperature (12) within safe limits
7/26/2016, 10:08:42 AM device data
msg : Object
{ "payload": 12, "_msgid": "afa359ef.505ca8" }
7/26/2016, 10:08:44 AM cpu status
msg.payload : string [35]
Temperature (12) within safe limits
7/26/2016, 10:08:44 AM device data
msg : Object
{ "payload": 12, "_msgid": "e590e06b.1a6f2" }
7/26/2016, 10:08:46 AM cpu status

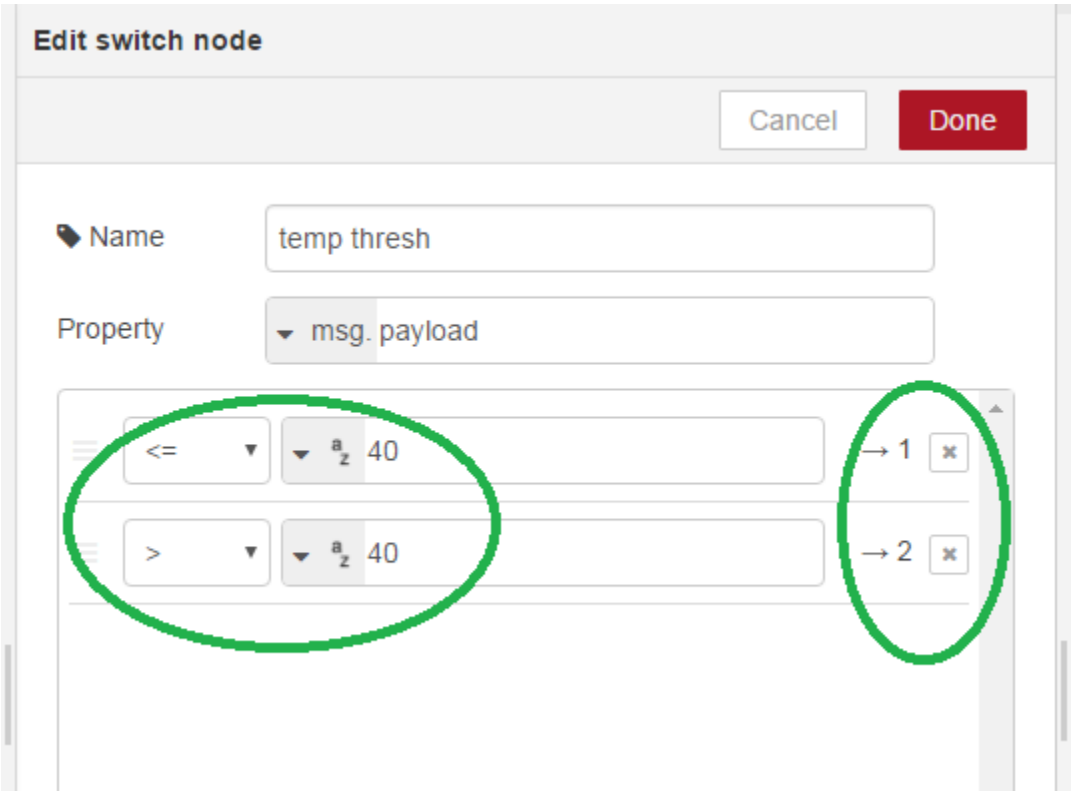


Note: If the network has latency issue then you might see some delay in viewing the data at the debug console but the data would be flowing to cloudant database. Hence proceed to store the data through the cloudant as discussed in the subsequent steps

Validate the node functionality

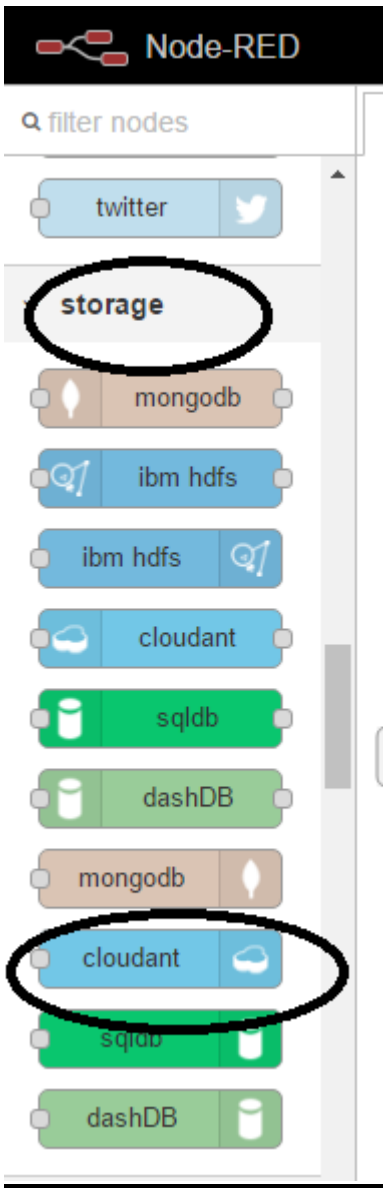
Increase the temperature at the simulator and validate the response from node.

The default node created has the safe limit as 40 and critical as >40. Change this and validate the response.

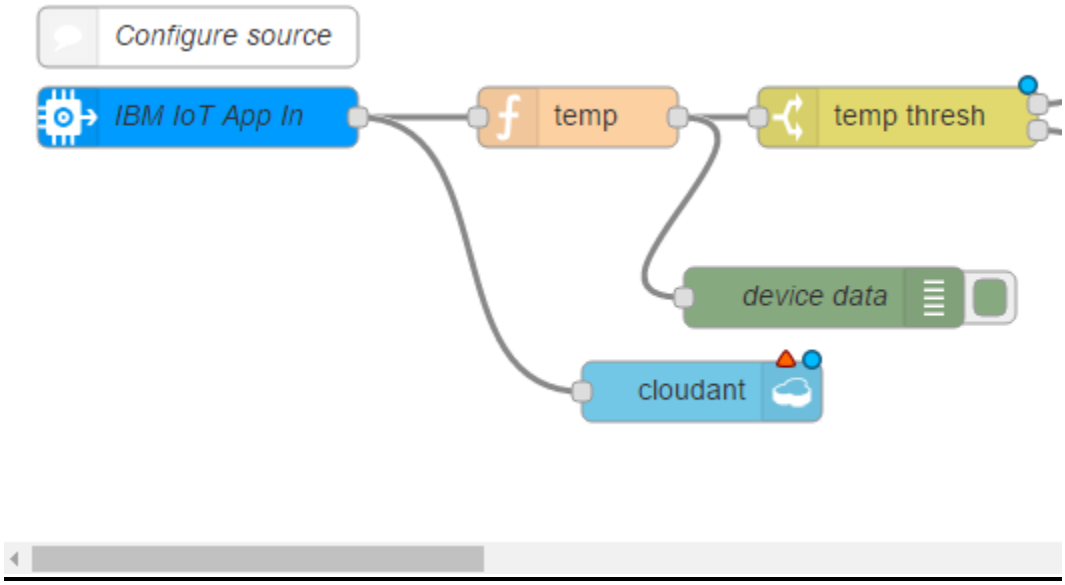


Store the data to cloudant

Select the cloudant node from the left panel.

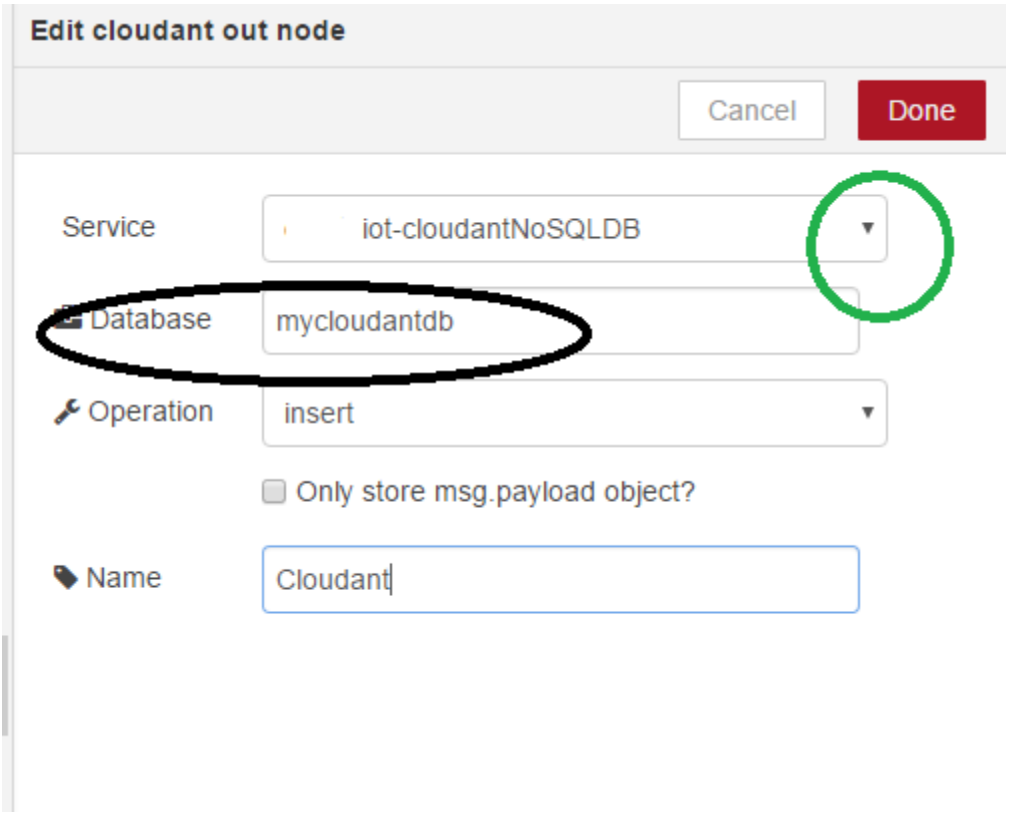

Node-RED
filter nodes
twitter
storage
mongodb
ibm hdfs
ibm hdfs
cloudant
sqldb
dashDB
mongodb
cloudant
sqldb
dashDB

Channel the output from the simulator to cloudant as shown below

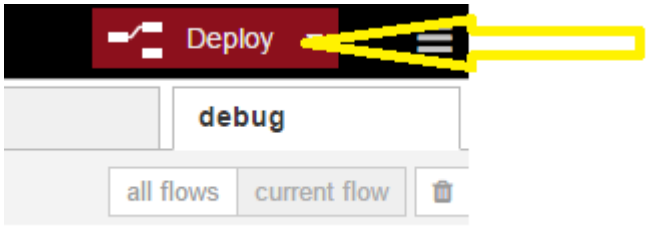


Provide the cloudant database details. Since the cloudant database is already part of the deployed application in Bluemix, the credentials are readily available.

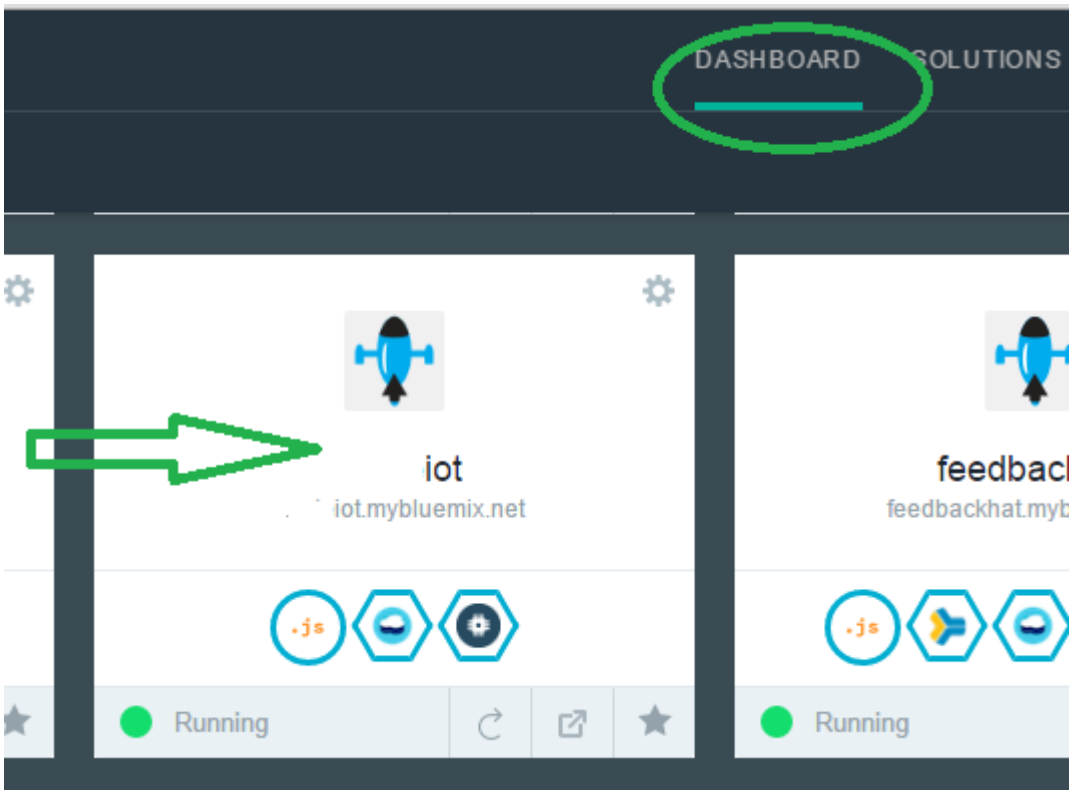
By default the database provisioned is listed. Provide the name for the database instance where the simulator data will be stored.



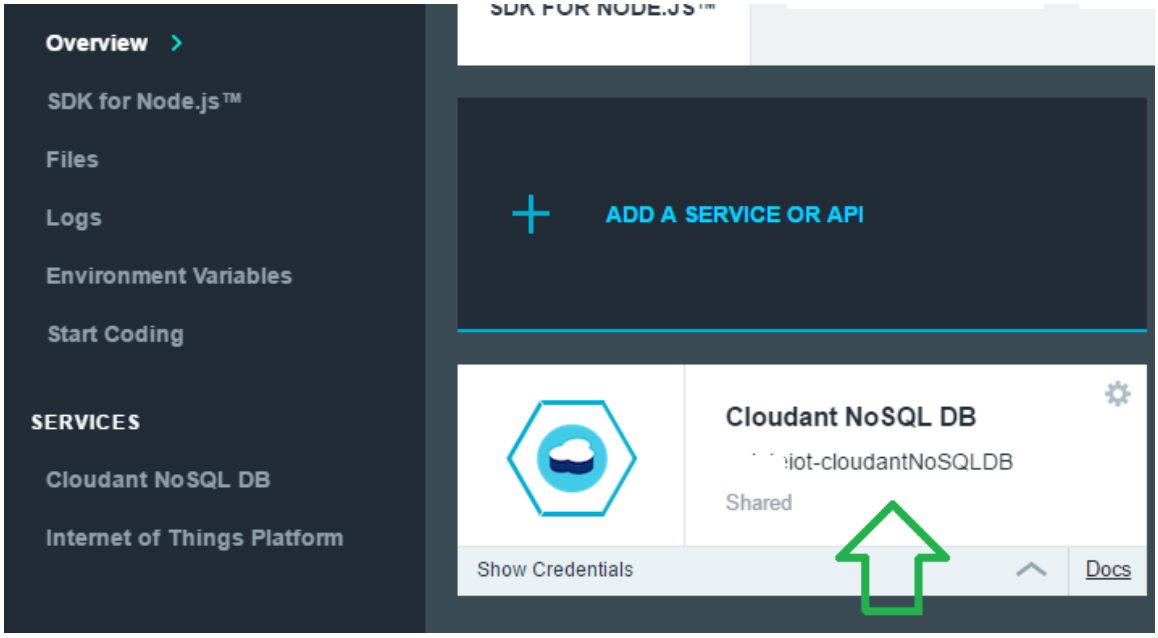
Deploy the node again:



To view the data, go back to the bluemix dashboard. Select the application deployed.

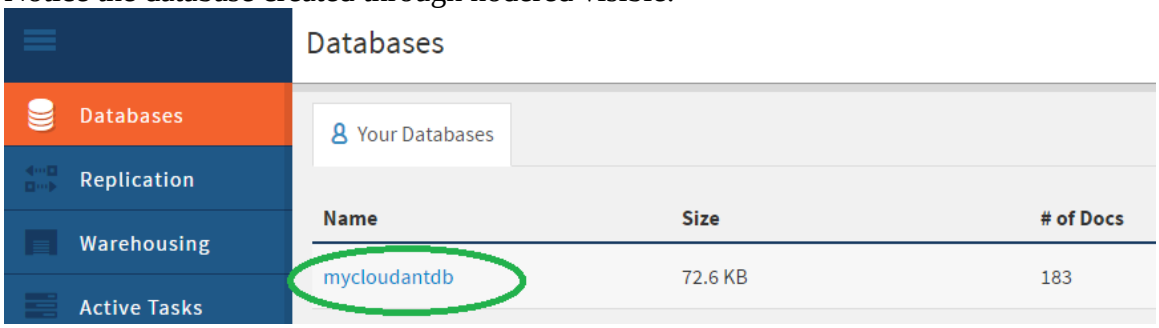


Select the cloudant database :



Launch the database console:

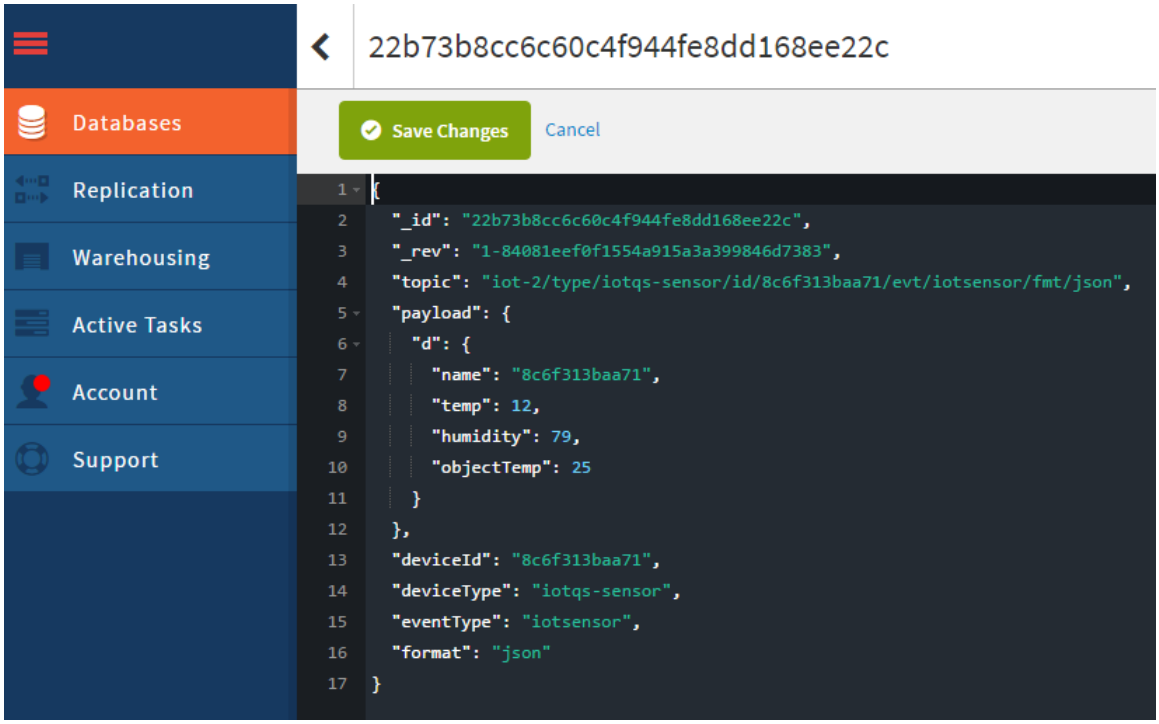
Notice the database created through nodered visible.



In the above case, already 183 documents are been created.

Now, go ahead and Click on the database “mycloudantdb” to view the data:

One of the data stored as document in the cloudant:

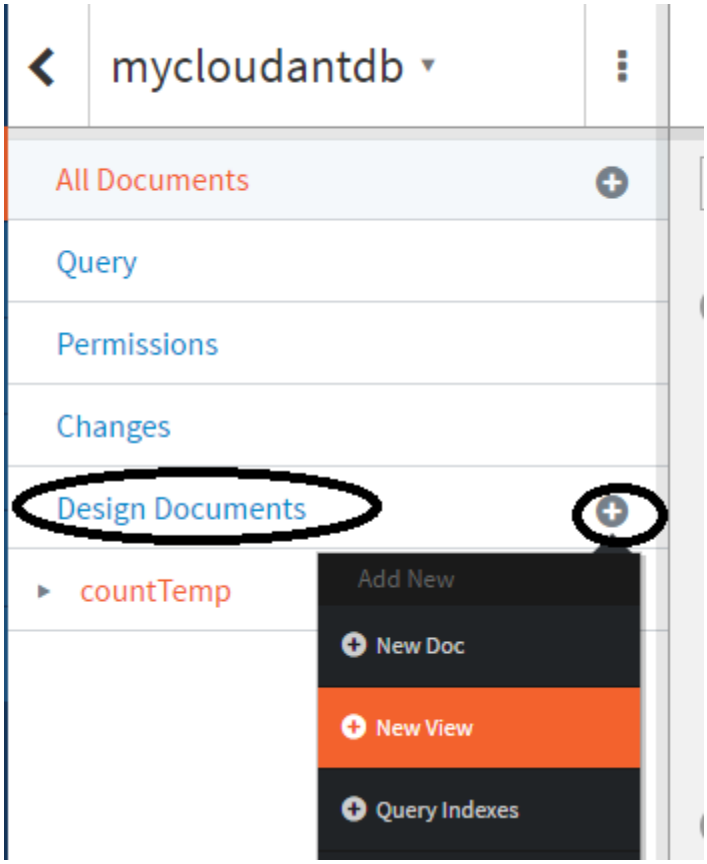


Build Query :

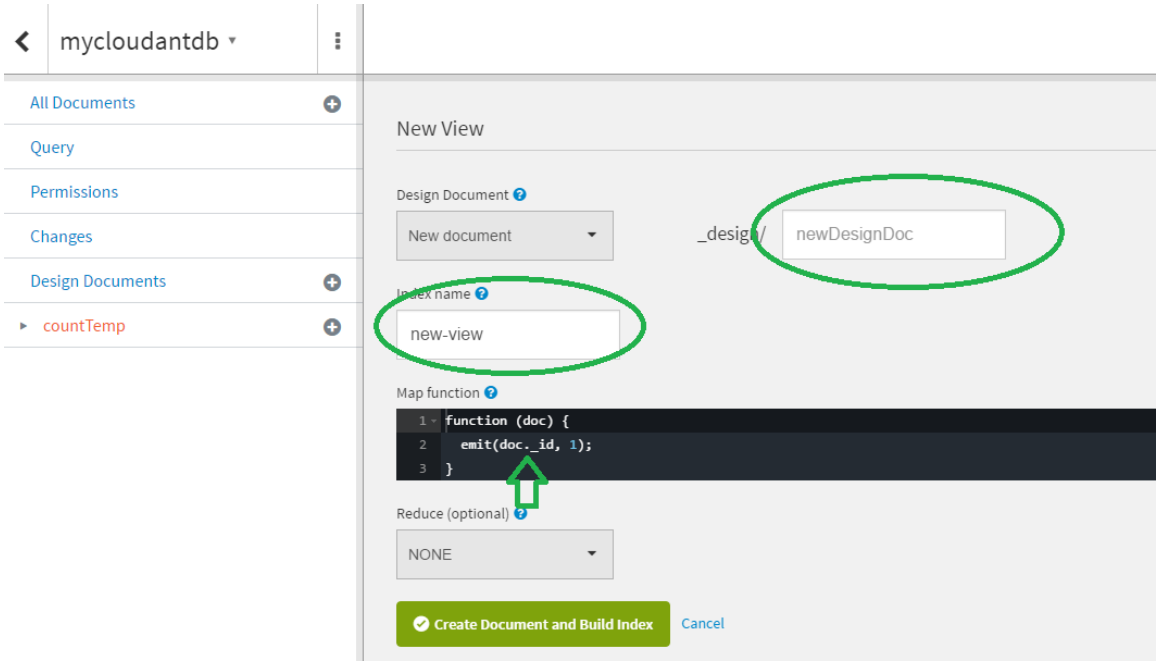
Requirement: count the entry of temperature data:

Note: cloudant provides views which uses map-reduce to list the data.

Click on “Design Documents” and build a new view as shown:



Provide the view name and Index name and write the map-reduce structure to retrieve the data:



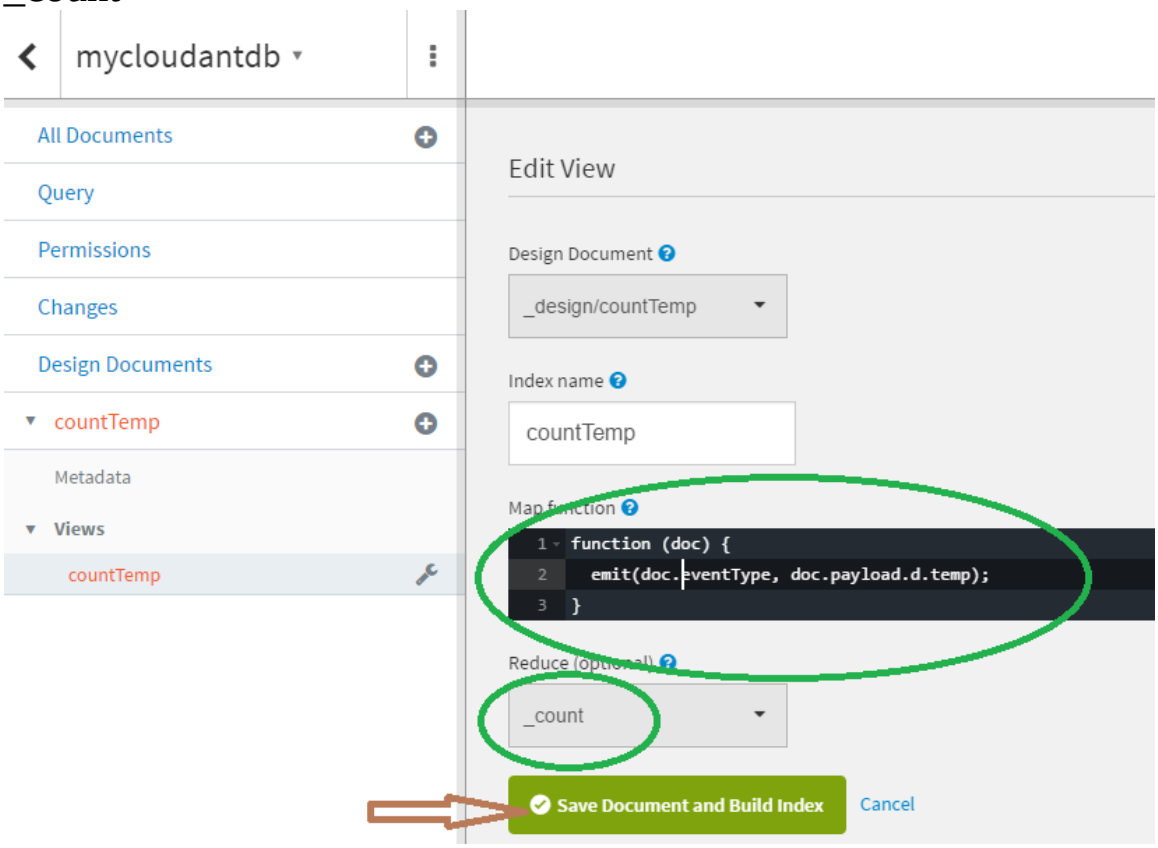
map

```
function (doc) {
  emit(doc.eventType, doc.payload.d.temp);
}
```

Note:
Doc.eventType -> **key**
Doc.payload.d.temp -> **value**

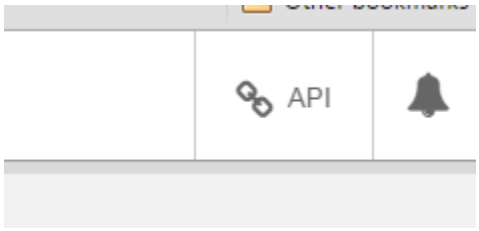
Reduce:

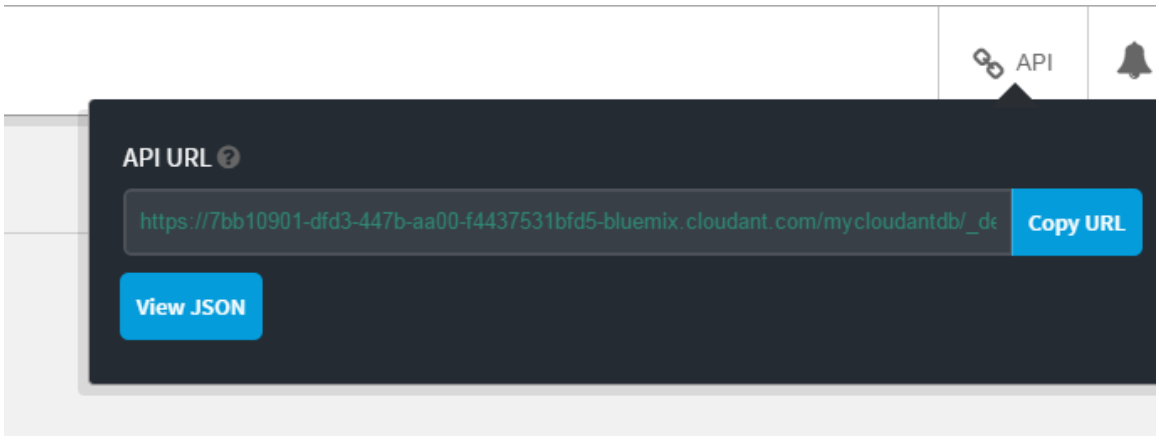
_count



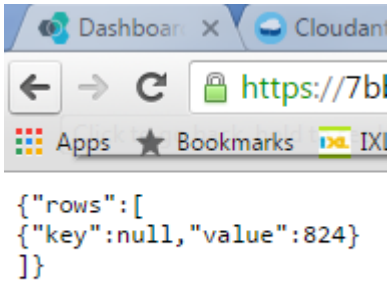
Save and build the view. You will get the instant output .

The output can also be viewed as api, which can be used in the web or the mobile application:





Copy the URL and enjoy viewing it in the browser or web or mobile applications..



In this case, we have 824 entries for temperature in the cloudant database.

Congrats !! you have complete the Bluemix Watson IoT cloudant Module

Exercise:

Build a query to list the count of temperature > 40 deg. (In this case, use your simulator to increase and decrease the temperature to +/- 40 and run the query against that data.

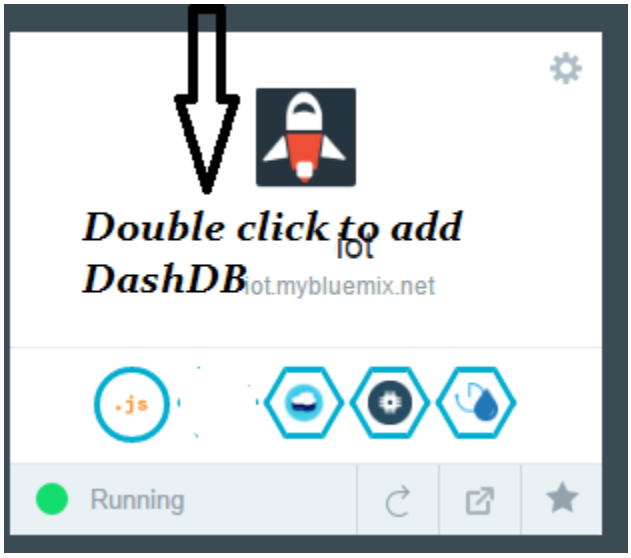
Hint:

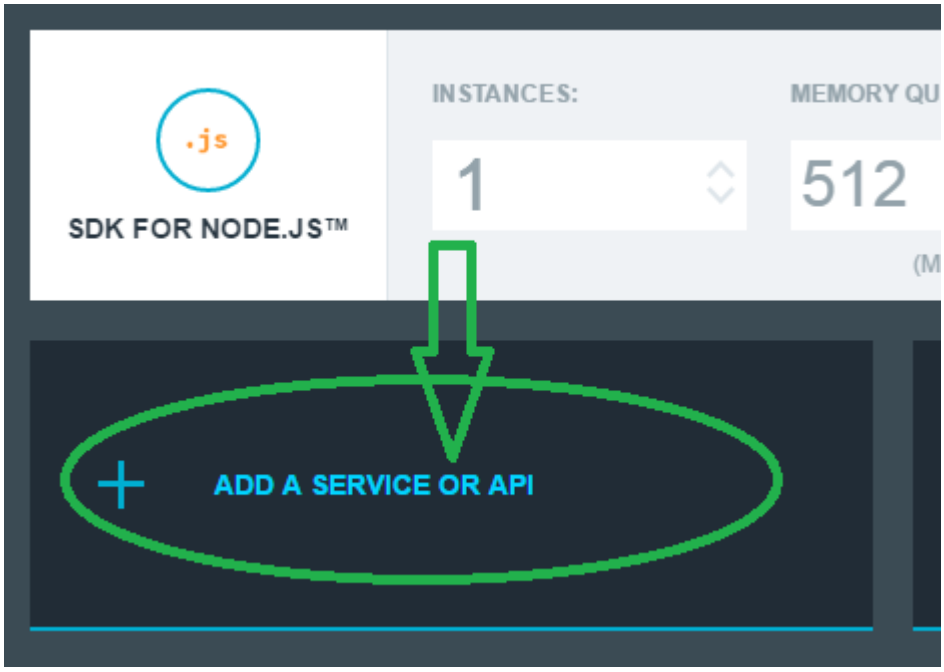
```
function (doc) {
  if (doc.payload.d.temp > 40)
  emit(doc.eventType, doc.payload.d.temp);
}
```

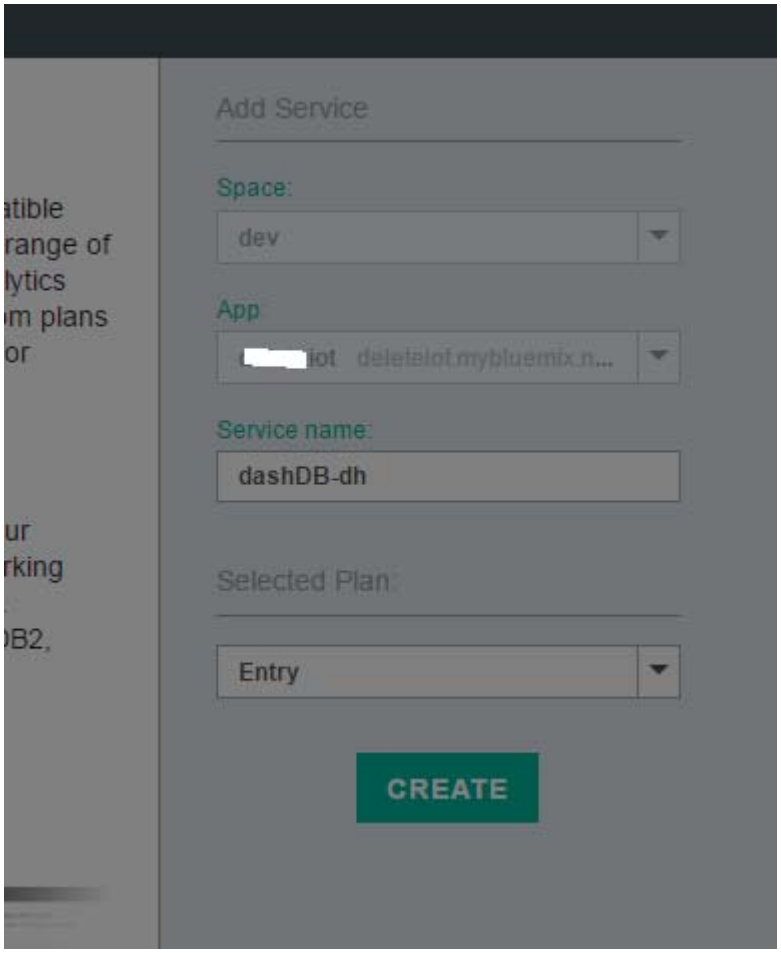
Lab 2: NO-SQL to SQL - Create a warehouse with SQL based DashDB for IoT data SQl query and R script plotting

Add a “dashDB” service through the IoT application got deployed

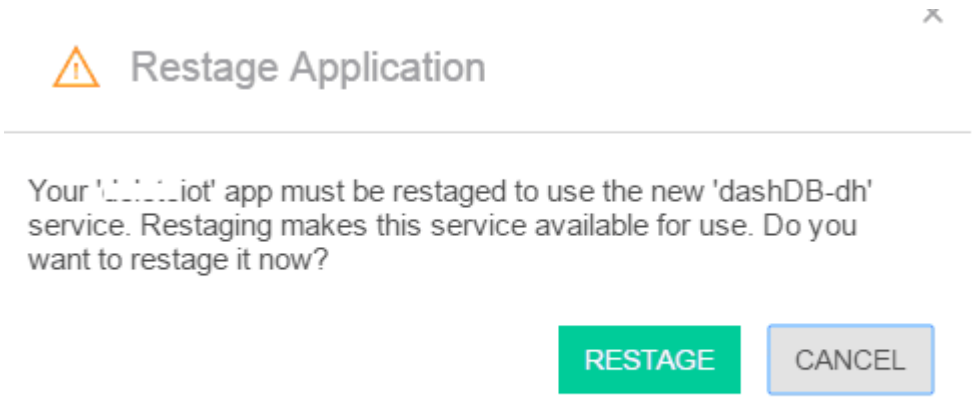
-> go to Bluemix dashboard and select the IoT application. Click on “Add a service” and bind the dashDB service



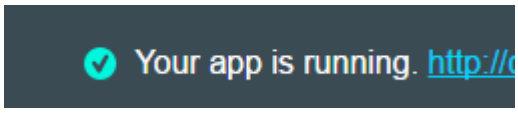


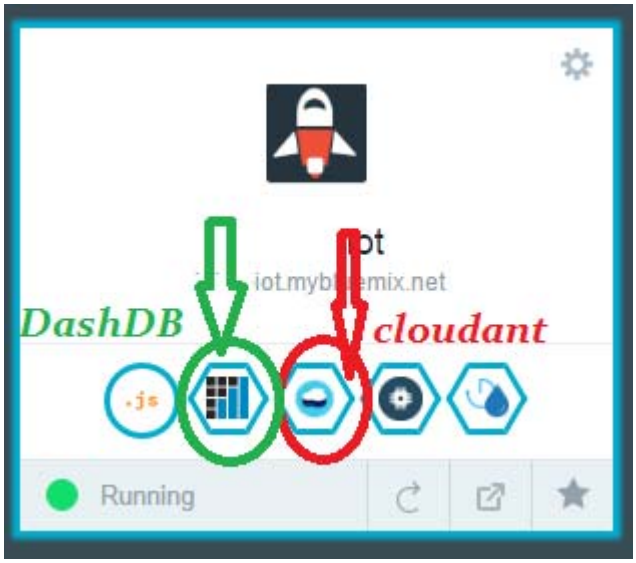


Allow the application to get restaged



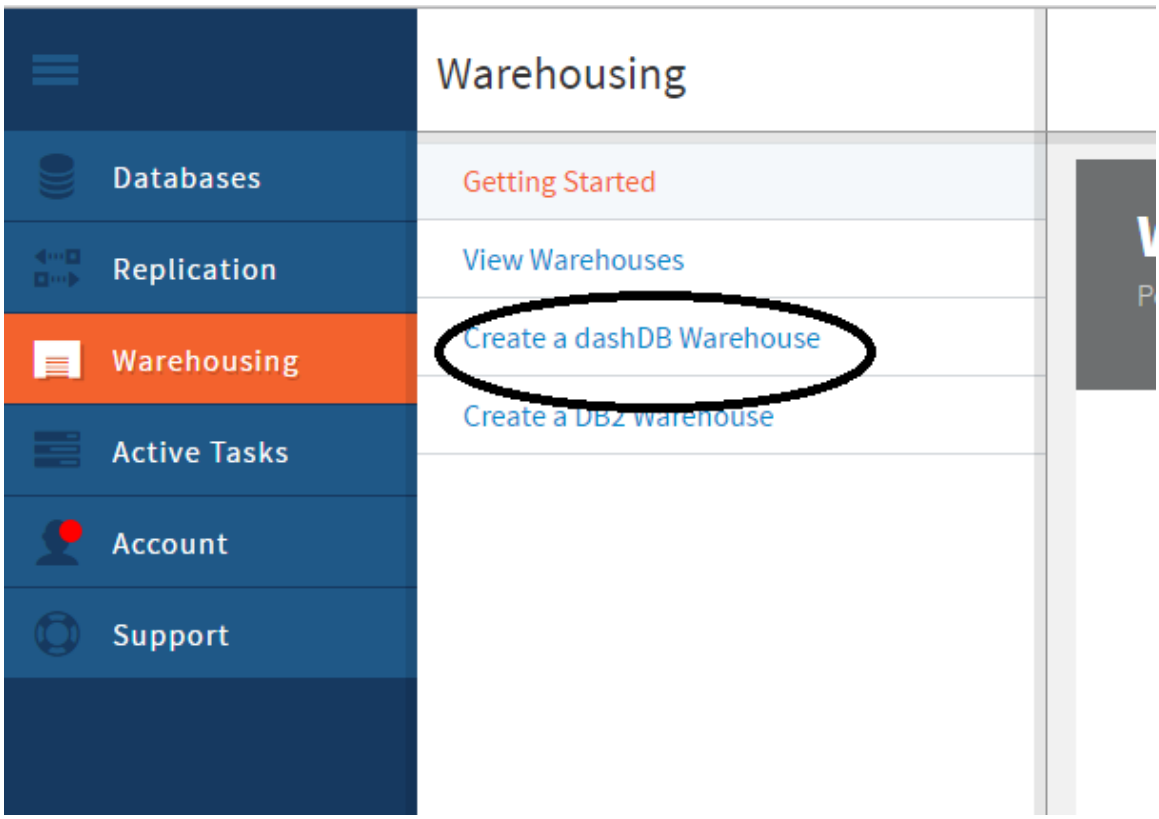
Wait till the application gets into running stage



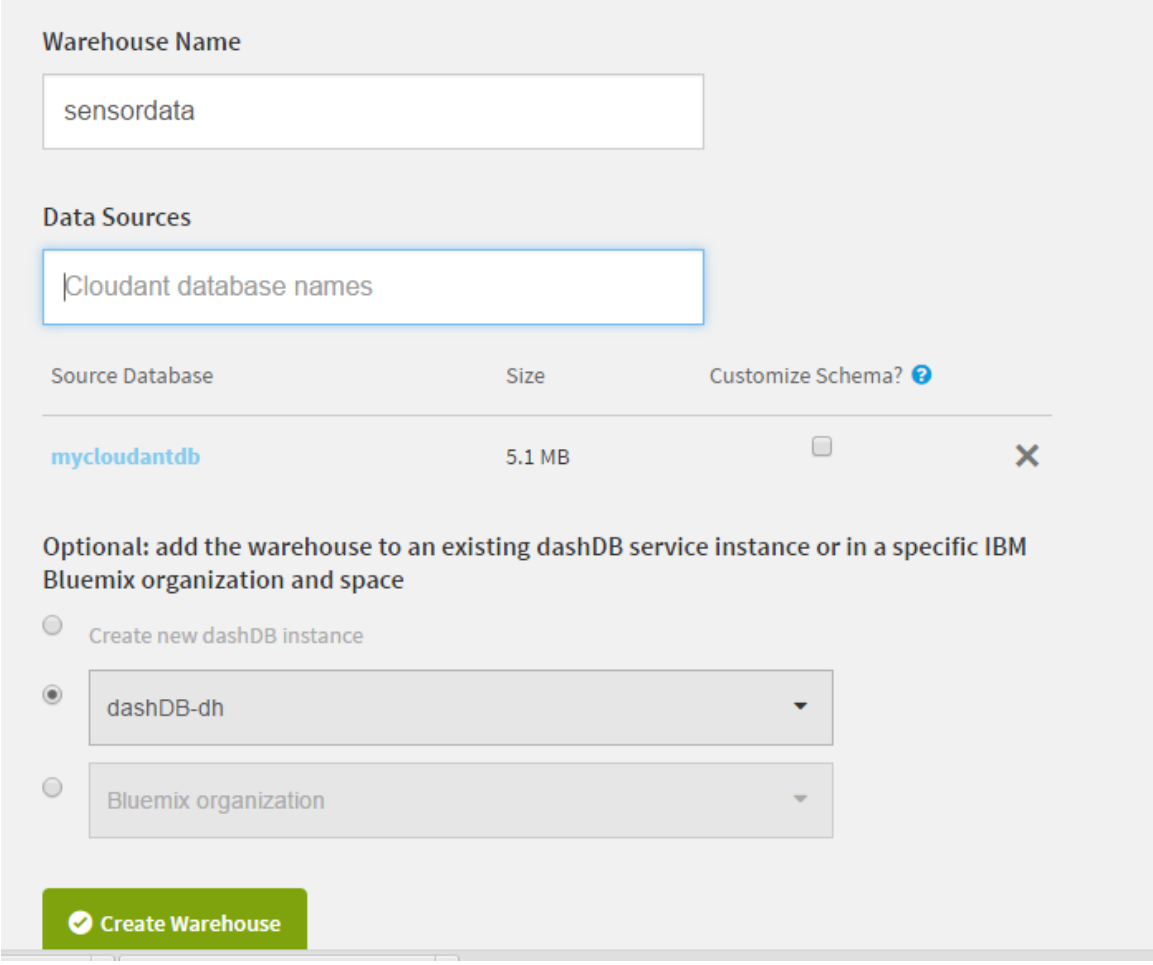


Go back to the dashboard and open the cloudant database console. Our objective is to create the dashDB warehouse through cloudant database. Through this we can convert the no-sql data to sql based data

Click on “Create DashDB” warehouse as shown

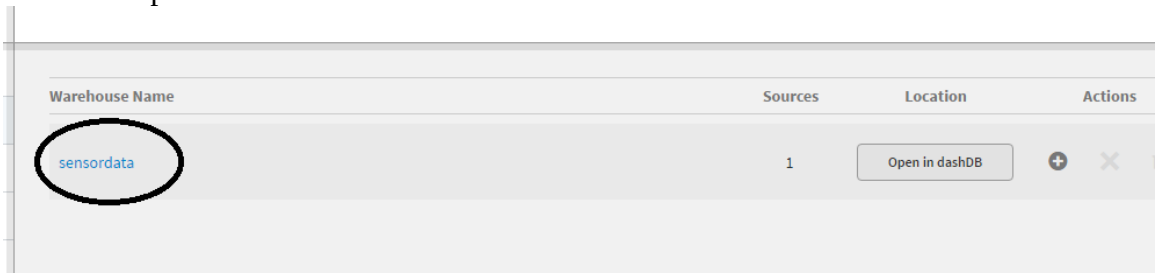


Provide the name for the warehouse and select the name of the cloudant database, in this case,
Warehouse name -> sensordata (provide your own unique name)
Cloudant Database -> mycloudantdb (select what you created as part of Lab1)



Post selecting the dashDB service as shown above, dashDB-dh , proceed to create warehouse.

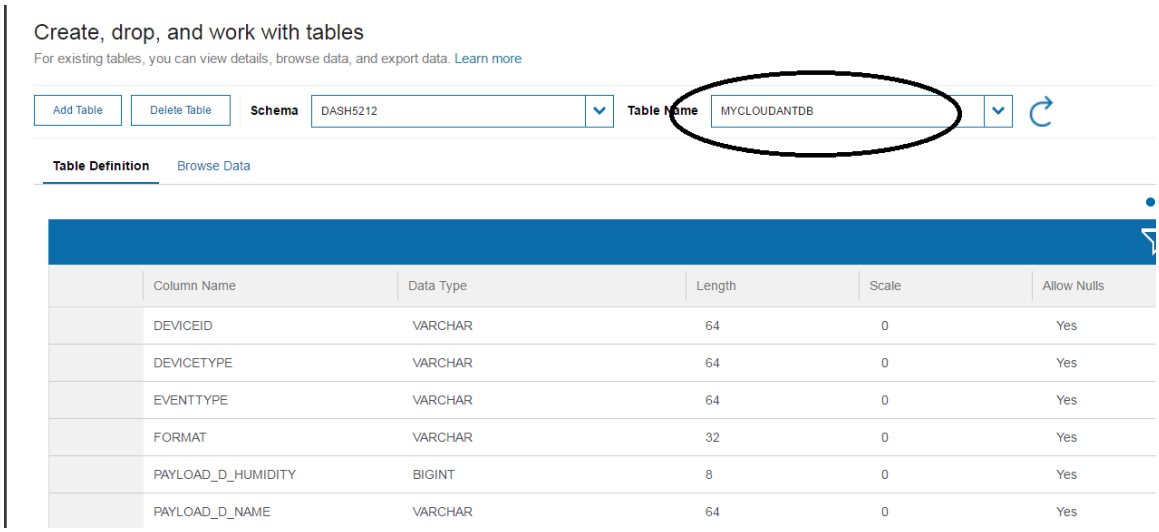
It takes couple of minutes. Post this the warehouse is been created as shown below



Congrats !! Now the sensor data is available in SQL format . Proceed to view this by selecting “Open in dashDB”

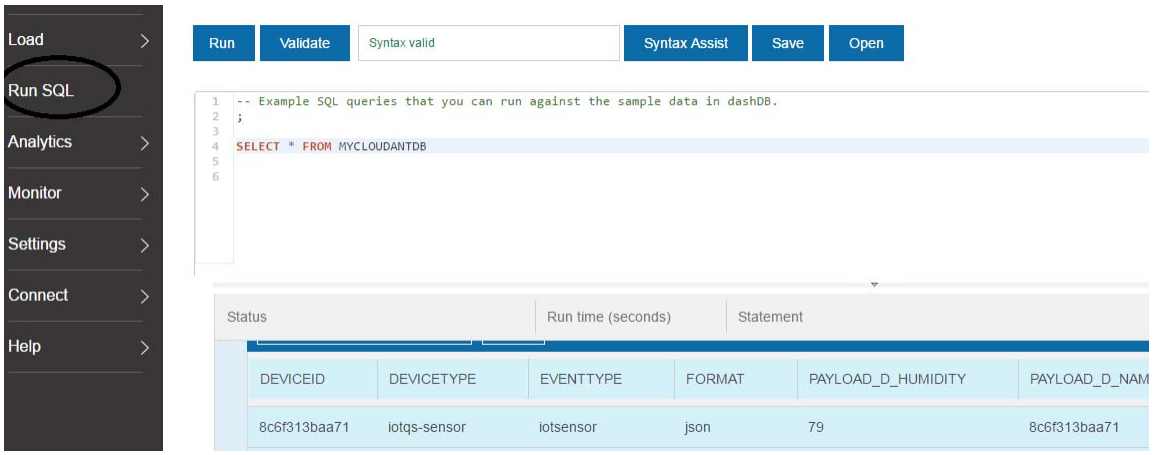
DashDB console gets opened as shown below.Procced to list the table content.

Under tables, load the table “mycloudantdb”, name of the the table created through cloudant



Run your favorite SQL query and enjoy viewing the sensor data

For example: select * from MYCLOUDANTDB

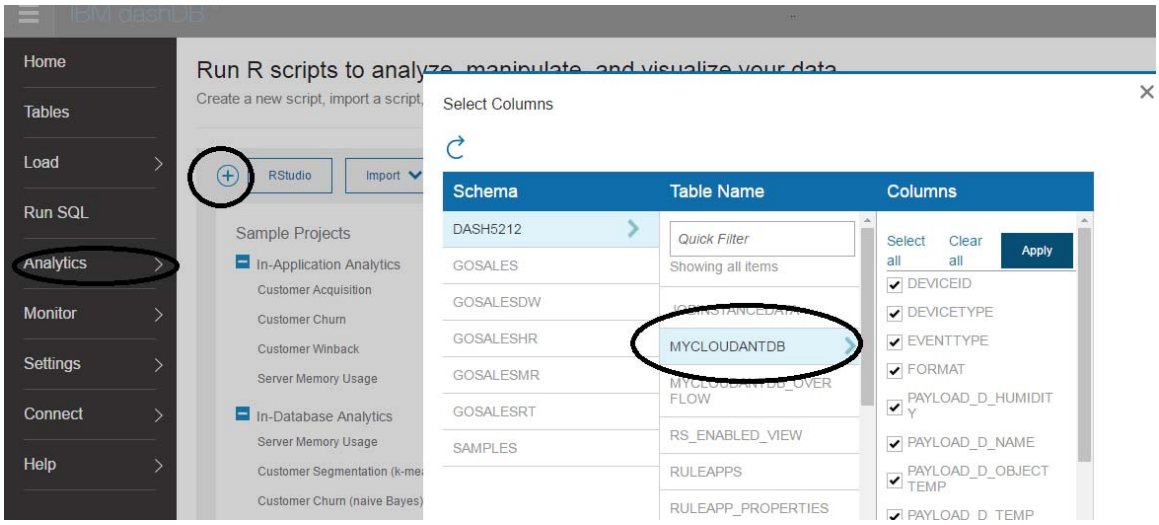


Quick plot using R script:

Click on Analytics and select the R script.

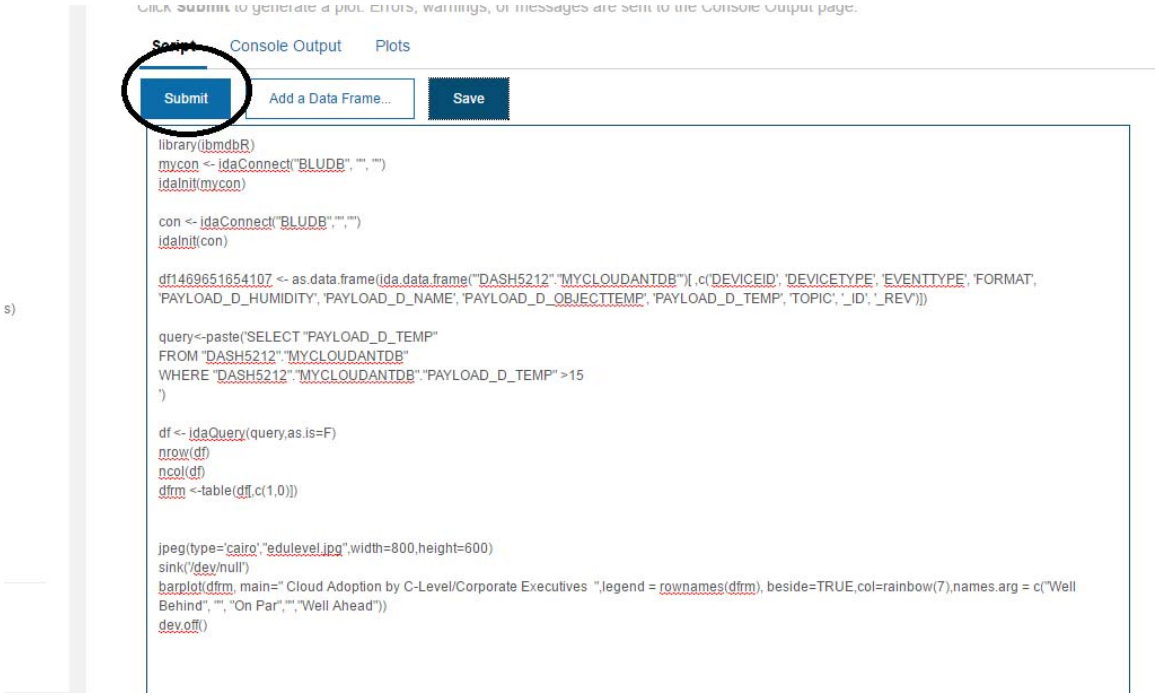
Proceed to create a new project by selecting the + symbol as shown below:

Select the table and the columns to create a dataset



Add the following sample script and save it.

Post saving ‘Submit” it to see the data is been plotted.

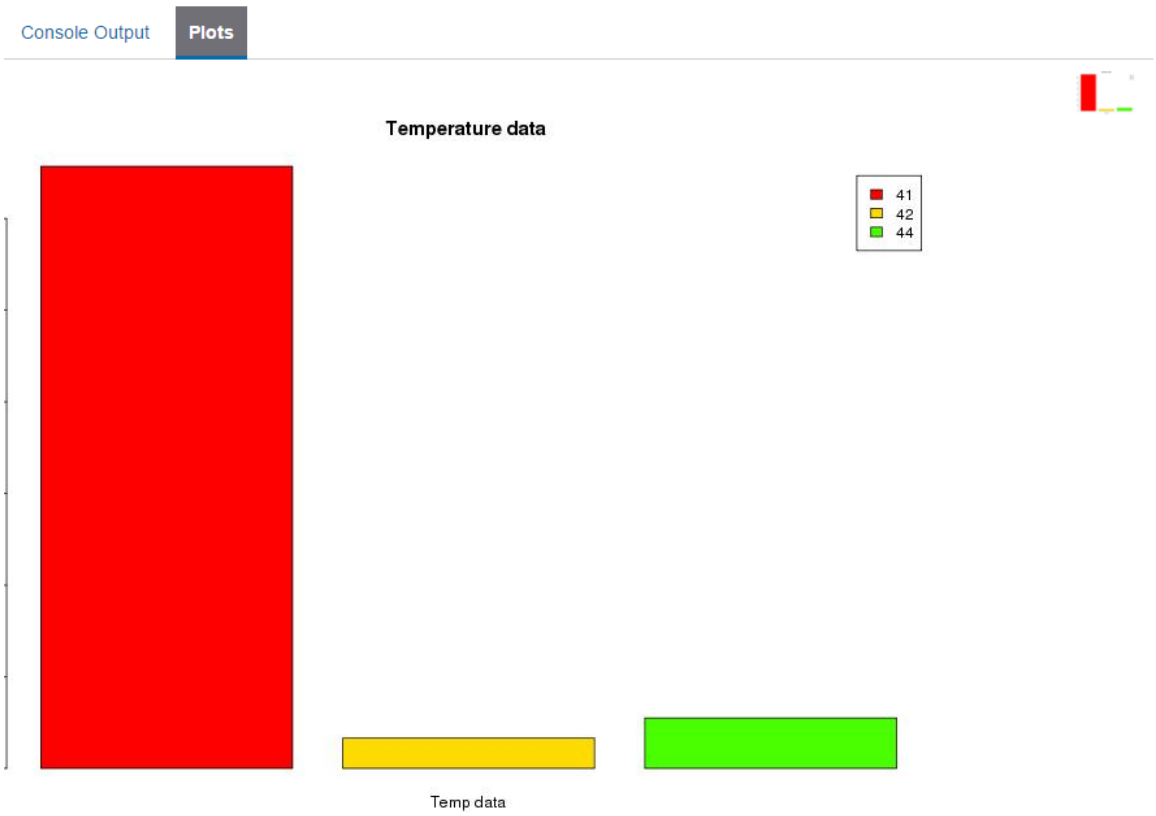


Sample Script:

/*** DO NOT COPY THE COMMENTED CODE since you will have your own data frame created:

```
library(ibmdbR)
mycon <- idaConnect("BLUDB", "", "")
```

```
idaInit(mycon)


df1469651654107 <-
as.data.frame(ida.data.frame('"DASH5212"."MYCLOUDANTDB"')[ ,c('DEVICEID',
'DEVICETYPE', 'EVENTTYPE', 'FORMAT', 'PAYLOAD_D_HUMIDITY',
'PAYLOAD_D_NAME', 'PAYLOAD_D_OBJECTTEMP', 'PAYLOAD_D_TEMP',
'TOPIC', '_ID', '_REV')])
**********/

/* COMMENT ->replace the schema (DASH5212) and table (MYCLOUDANTDB) as
per the details what you have …*/

con <- idaConnect("BLUDB","","")
idaInit(con)

query<-paste('SELECT "PAYLOAD_D_TEMP"
FROM "DASH5212"."MYCLOUDANTDB"
WHERE "DASH5212"."MYCLOUDANTDB"."PAYLOAD_D_TEMP" >40
')

df <- idaQuery(query,as.is=F)
nrow(df)
ncol(df)
dfrm <-table(df[,c(1,0)])


jpeg(type='cairo',"templ.jpg",width=800,height=600)
sink('/dev/null')
barplot(dfrm, main=" Temperature data  ",legend = rownames(dfrm),
beside=TRUE,col=rainbow(7),names.arg = c("Temp data"))
dev.off()
```



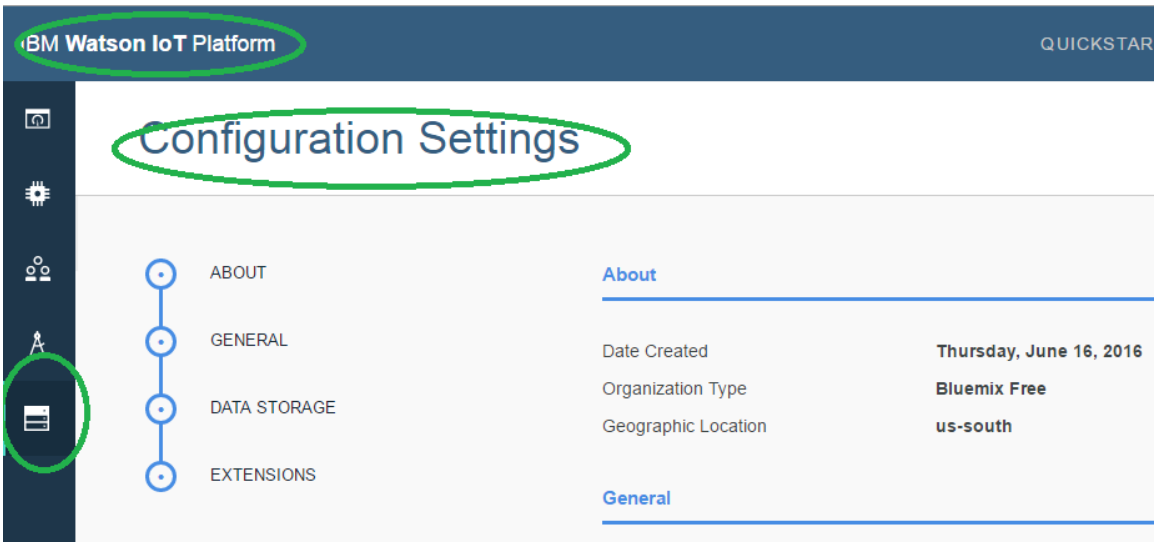
Congrats !! You have completed the Lab2. At this stage you have learned to play around the IoT data using the SQL syntax and then viewing it using R script

Exercise:

Create the data monitoring using Jasper framework and use it in the IoT Platform to view it

Hint:
IoT platform provides option to view the jasper framework as shown below:

- ➔ go to IoT platform and select “configuration setting”



➔ scroll down to ‘Extensions” and provide the jasper credentials to use your template