IBM Bluemix & Watson Labs

April 2017

Benoit Marolleau
Architect – IBM Montpellier Client Center
IBM France
benoit.marolleau@fr.ibm.com

#bluemix

http://bluemix-watson-day.mybluemix.net

Agenda

- ✓ Lab 1 – Create & Deploy a simple App in Bluemix
- ✓ Lab 2 – Get Started with Bluemix DevOps
- ✓ Lab 3 – IoT Application with Node-RED and IBM Watson Services
- ✓ Lab 4 – Watson & Face Recognition
- ✓ Lab 5 – Watson & Conversation
- ✓ Optional Labs

Labs – Before Starting

- ✓ Access to Bluemix (requires a valid Bluemix Account)
  https://console.ng.bluemix.net

- ✓ You can use a free 30-day trial account or follow the HOW-TO guide (Academic Initiative for Cloud) for free extended capabilities.

- ✓ Free resources (GB / #Services) in your Bluemix Organization / Spaces to run the lab exercises.
  For this hands–on session, you will have to use US-South region to have access to required services (latest versions of services)
  If you encounter a resource contention (Error Message saying you are out of resources), clean up your Spaces by deleting existing Apps or Services.

- ✓ Bluemix Public is used for this Lab. Check Bluemix Status & Maintenance schedule on https://status.ng.bluemix.net/

Lab 1 – Create & Deploy a Simple Application

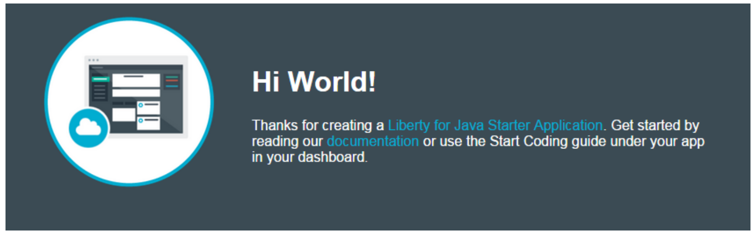
Bluemix Console – Dashboard - Service Catalog – Binding – Logs – Service Status

IBM

#bluemix

Lab 1 – Objective

- Connect to Bluemix, browse the Service Catalog
- Create your first Web Application in Bluemix
- Get familiar with Bluemix Graphical Interface & your Dashboard
- Expected Result: your Web App is operational

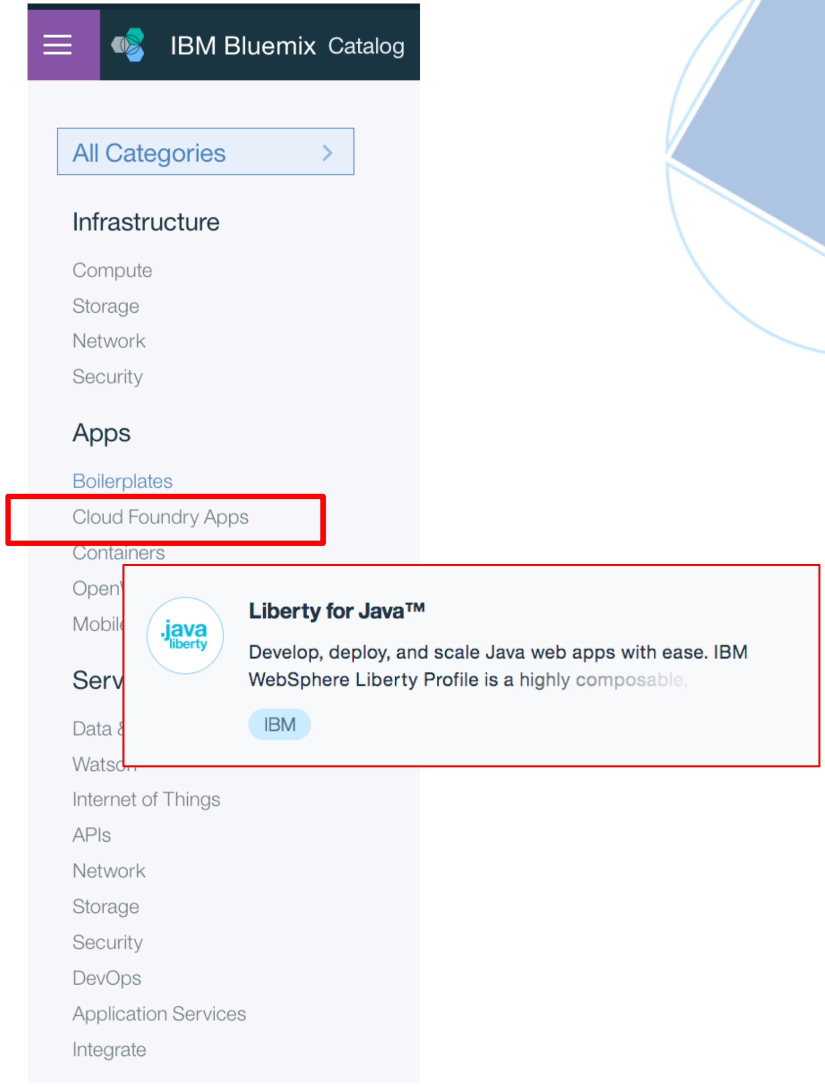


Hi World!

Thanks for creating a Liberty for Java Starter Application. Get started by reading our documentation or use the Start Coding guide under your app in your dashboard.

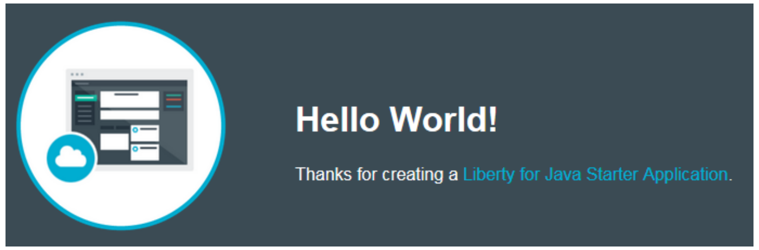
Lab 1 – Create your first App

1. Go to Bluemix https://console.ng.bluemix.net/ & log in
2. See on the top right : Name of your organization & Region (default: US South = IBM Cloud Dallas Datacenter)
3. Go to "Catalog" & scroll down. On the left side, see the categories for a direct & faster access to particular services or runtimes.
4. In the "Cloud Foundry Apps" categories, choose "Liberty for Java" for hosting a Java Enterprise application.
5. Enter the App Name, Host name (part of the domain name to access your app – has to be unique), and click on the CREATE button.
6. Wait for the app to be created. While the container is being created, have a look to the current documentation page in your DASHBOARD, The Overview page explains how to use the CF (cloud foundry) client to push your application code to Bluemix from your desktop. *It is not part of this lab session. An optional lab is available if you want to test the CF cli or Eclipse CF Plugin*.
7. Your App is now in a "Running" state (green) Click on the "VIEW APP" button on the right.
8. On the previous page (Dashboard), go to left side: "Overview" , "Runtime", "Connections", "Logs"



Liberty for Java™

Develop, deploy, and scale Java web apps with ease. IBM WebSphere Liberty Profile is a highly composable,



Hello World!

Thanks for creating a Liberty for Java Starter Application.

Lab 1 – Optional: Create & Bind a Service

Note : “Connecting” or binding a [Watson or anything] service to an App is not mandatory to use this Service (a service can run without an application). It is just an elegant way to get the service information instead of hardcoding it in the app. This information can be used by your application code to access the service. Service information can be: URL, user, password, port, IP …. Depending on the type of service. For example, a typical Watson service = your Service URL + Credentials

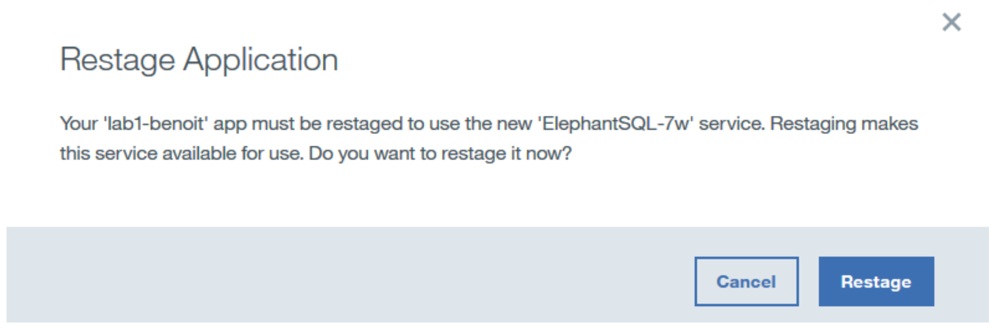
9. Let’s add a new service & bind it to this application. For example, let say that code newly Java application will access a PostgreSQL Database . In “Overview” > “Connections” , “Connect New”

10. Choose the appropriate service. In our case, ElephantSQL (third party “PostgreSQL as a service Service). Name your service, Choose the FREE plan (<20MB of Data) and click CREATE.
This will create your Database service and bind it to your application (The GUI will perform two CF commands for you). Note: Binding ⇔ Create environment variables that your app can use to access the bound services. This creation is done by a “Restage Application”.

11. Restage your application as requested

12. Click on “View Connection” to see the Database service information.

13. Go to your App Dashboard “Runtime” panel > “Environment Variables”.
In our simple “Hello World” App, this does not change anything as we have not modified our code for using this newly created service.


Restage Application

Your 'lab1-benoit' app must be restaged to use the new 'ElephantSQL-7w' service. Restaging makes this service available for use. Do you want to restage it now?

Lab 2 – Get Started with Bluemix DevOps

Cloud Foundry – Bluemix DevOps – Application Coding, Sharing & Deployment

#bluemix

Lab 2 – DevOps basics

1. Go to the following IBM Cloud Garage Method Tutorial Web Page

   https://www.ibm.com/devops/method/tutorials/tutorial_toolchain_flow

Within a few minutes, you can create an open toolchain and start using it to continuously deliver a "Hello World" app in an IBM® Bluemix® environment.
In this tutorial, you create a toolchain from a template that contains a specific set of tool integrations and code to develop and deploy a sample Cloud Foundry app that is written in Node.js.
The toolchain is preconfigured for continuous delivery, source control, issue tracking, and online editing.
After you create the toolchain, you change the app's code and push the change to the GitHub repository (repo).
When you push changes to your repo, the delivery pipeline automatically builds and deploys the code that is in the repo.

(If needed, PDF file is provided)

Lab3 – IoT Application with Node-RED and IBM Watson Services

IoT – Node-RED – Boilerplates – Watson Services

#bluemix

Lab 3 - Objectives

- Create & modify an application using Node-RED

- Discover new services (IoT) & Node-RED, a visual tool (Open source project developed by IBM) to easily develop JavaScript applications, consume or create services (IoT / Watson…)

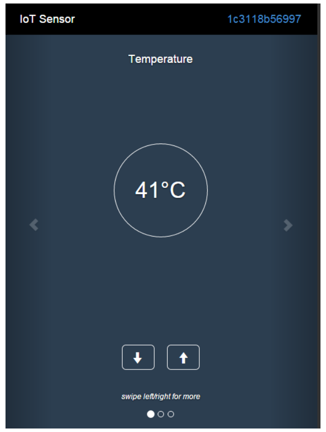
Lab 3 – Expected Results

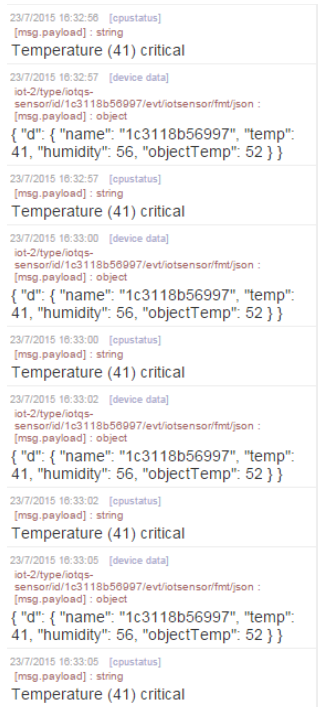
Your Node-RED application is operationnal (using Node.js runtime), accessing Cloudant & IoT Foundation Services (QuickStart)

Your App is online (reachable from the Internet), & will be connected to a temperature simulator (sensor)

Prerequisites - Download the JSON file:

http://bluemix-watson-day.mybluemix.net/files/Lab3-bluemix-iot.json

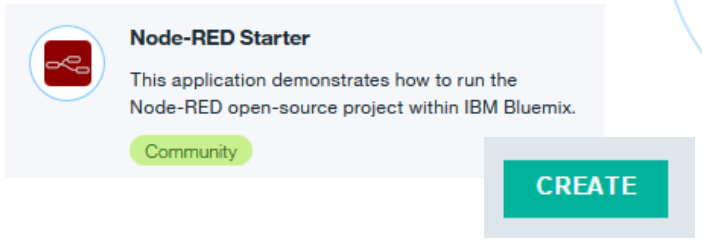




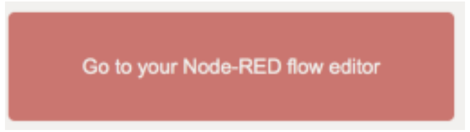
```
23/7/2015 16:32:56   [cpustatus]
 [msg.payload] : string
Temperature (41) critical
23/7/2015 16:32:57   [device data]
 iot-2/type/iotqs-
 sensor/id/1c3118b56997/evt/iotsensor/fmt/json :
 [msg.payload] : object
{ "d": { "name": "1c3118b56997", "temp":
41, "humidity": 56, "objectTemp": 52 } }
23/7/2015 16:32:57   [cpustatus]
 [msg.payload] : string
Temperature (41) critical
23/7/2015 16:33:00   [device data]
 iot-2/type/iotqs-
 sensor/id/1c3118b56997/evt/iotsensor/fmt/json :
 [msg.payload] : object
{ "d": { "name": "1c3118b56997", "temp":
41, "humidity": 56, "objectTemp": 52 } }
23/7/2015 16:33:00   [cpustatus]
 [msg.payload] : string
Temperature (41) critical
23/7/2015 16:33:02   [device data]
 iot-2/type/iotqs-
 sensor/id/1c3118b56997/evt/iotsensor/fmt/json :
 [msg.payload] : object
{ "d": { "name": "1c3118b56997", "temp":
41, "humidity": 56, "objectTemp": 52 } }
23/7/2015 16:33:02   [cpustatus]
 [msg.payload] : string
Temperature (41) critical
23/7/2015 16:33:05   [device data]
 iot-2/type/iotqs-
 sensor/id/1c3118b56997/evt/iotsensor/fmt/json :
 [msg.payload] : object
{ "d": { "name": "1c3118b56997", "temp":
41, "humidity": 56, "objectTemp": 52 } }
23/7/2015 16:33:05   [cpustatus]
 [msg.payload] : string
Temperature (41) critical
```

Lab 3 – IoT & Node-RED – Create a new Flow

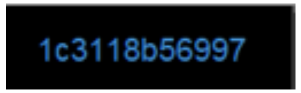
1. In Bluemix Catalog, Choose "boilerplate" Node-RED Starter & create an instance: CREATE button. Fill in the App Name & host Name fields.
Note: Node-RED is a Node.js based application: using this boilerplate will instantiate a Node.js runtime + a Cloudant (NoSQL DB) service.
Click Create. Wait for the env to be created & the App to start (~4 minutes).


Node-RED Starter
This application demonstrates how to run the Node-RED open-source project within IBM Bluemix.
Community
CREATE

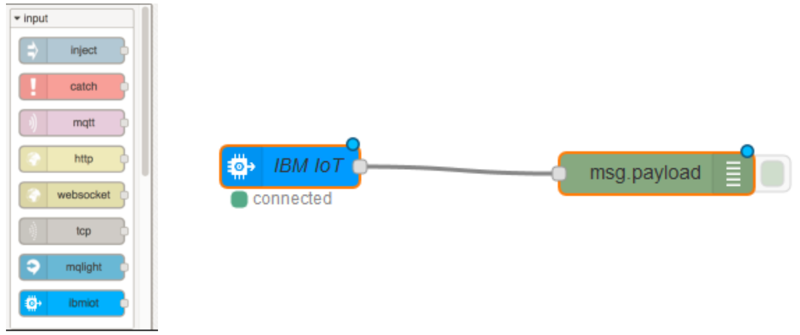
2. Access the Node-RED application (View App) & click the Red button to launch the Node-RED Flow Editor.



3. Sensors & IoT - Create a simulator & identify your device ID (top right corner).
http://ibm.biz/iotsensor *Note (Optional) Instead of using your desktop browser, use your smartphone!*



4. In the Flow Editor, Create a Flow (drag & drop of boxes on the left panel Chose the Input node « IBM IoT ». Add an output « Debug » node & link them.



5. Configure « IBM IoT » by Double clicking on it :

- Authentication : Quickstart (means it is a simple authentication – for demo purposes)
- Device ID : <The value from Step 3 - Generated by the Simulator>

6. Click Done & Deploy your flow by clicking the « Deploy » button (top right).

Lab 3 – IoT & Node-RED – Import a Flow

7. Check the Debug Panel on the right side while you are playing with the sensor simulator. You should receive Device (sensor) data as the IBM IoT Node subscribed to this particular Device topic.

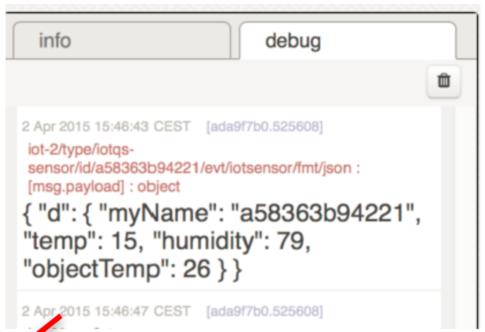
8. Delete the whole Flow by selecting all the nodes & pressing the 'Delete' key.

9. Create a new flow – This time – by importing the code (URL : http://bluemix-watson-day.mybluemix.net/files/Lab3-bluemix-iot.json)

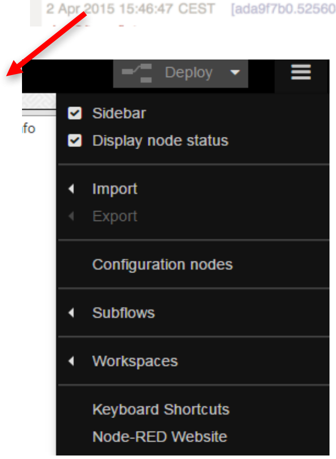
- Click on the top right button near Deploy.
- Select import, Clipboard & copy/paste the content of the JSON file

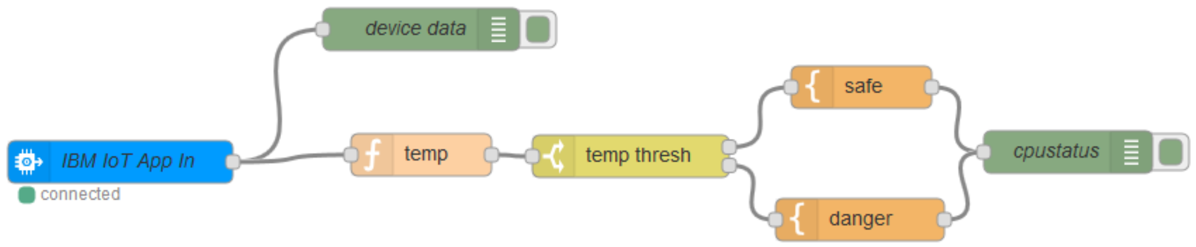
10. Deploy the new Flow. Fill in the Device ID field in the 'IBM IoT App In' Node.

Modify the Device Temperature & check the Debug logs.



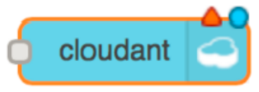
```
2 Apr 2015 15:46:43 CEST   [ada9f7b0.525608]
iot-2/type/iotqs-
sensor/id/a58363b94221/evt/iotsensor/fmt/json :
[msg.payload] : object
{ "d": { "myName": "a58363b94221",
"temp": 15, "humidity": 79,
"objectTemp": 26 } }
2 Apr 2015 15:46:47 CEST   [ada9f7b0.525608]
```



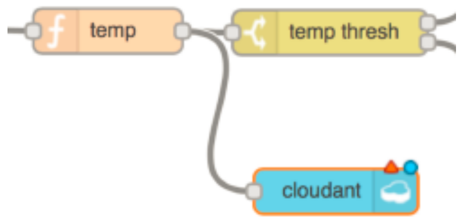


Lab 3 – IoT & Node-RED – Insert IoT Data in Cloudant DB

Let’s insert the event data coming from the Device sensors in a cloudant Database!

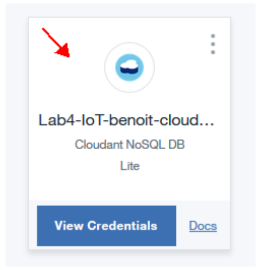


11. Add a Cloudant Node (output node in the Storage Category) & link it to the « Temp » function node See picture on the right =>



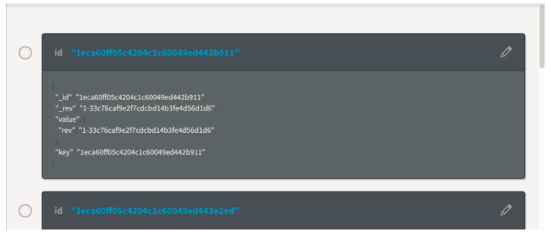
12. Configure it:

- Service : Cloudant service name bound to your Node.js runtime.
  As Node.js is already bound to a Cloudant Service, the service name should appear in the Drop-down list.
- Database : name of your choice (lower case)
- Name (node) : nom of your choice



13. Deploy your new flow

14. From your Bluemix Dashboard, start the Cloudant console, and have a look to the inserted data in the Database (name specified in step 12).

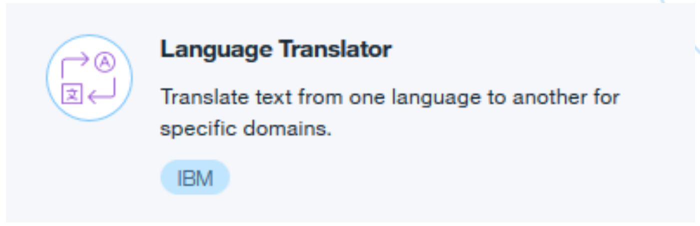


```
id "1eca60ff05c4204c1c60049ed442b911"
{
 "_id": "1eca60ff05c4204c1c60049ed442b911",
 "_rev": "1-33c76caf9e2f7cdcbd14b3fe4d56d1d6",
 "value": {
  "rev": "1-33c76caf9e2f7cdcbd14b3fe4d56d1d6"
 },
 "key": "1eca60ff05c4204c1c60049ed442b911"
}
id "1eca60ff05c4204c1c60049ed443e2ed"
```

Lab 3 – IoT & Node-RED – Process IoT Data with Watson

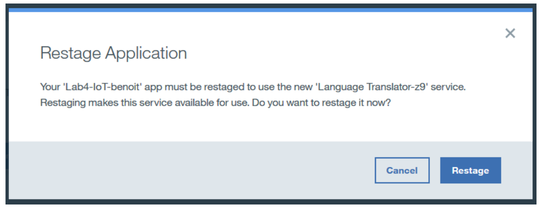
14. Add a « Watson Language Translation » service to your existing Node.js / Node-RED application and accept the Restage step to actually bind the service to the app.
> App Dashboard > Connections > Connect New, then click CREATE

Note: while it is restaging, go to Credentials : This information are useful if you want to invoke your Watson Service from any program (running in Bluemix or outside Bluemix)



Language Translator

Translate text from one language to another for specific domains.



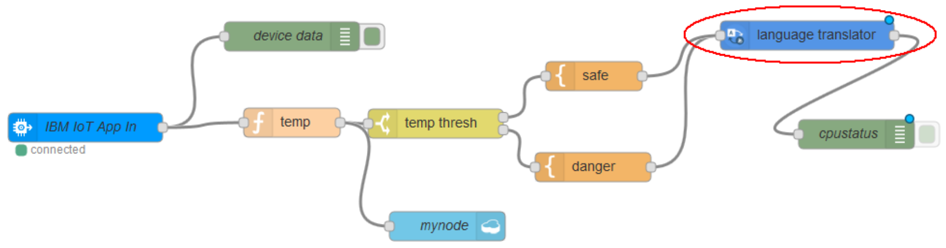
Restage Application

Your 'Lab4-IoT-benoit' app must be restaged to use the new 'Language Translator-z9' service. Restaging makes this service available for use. Do you want to restage it now?

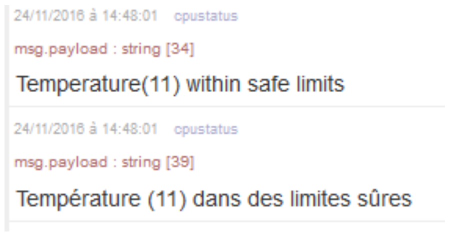
15. In Node-RED, add a « Watson language translation » node and link it between the template (*safe & danger) & debug cpustatus* nodes.

Configure the Watson language translator node:

- Name (of your node)
- Input language: English
- Output language: French
- Note:
  The user/password fields are not necessary & do not appear in the node settings if a Watson Translator service is properly bound to Node.js.!!!



16. Deploy your flow & check the logs!!



```
24/11/2016 à 14:48:01   cpustatus
msg.payload : string [34]
Temperature(11) within safe limits
24/11/2016 à 14:48:01   cpustatus
msg.payload : string [39]
Température (11) dans des limites sûres
```

Lab 4 – IBM Watson & Face Recognition

IBM Watson – Face Recognition with Visual Recognition – Node-RED

#bluemix

Lab 4 – Instructions

Build a Face Recognition App Using AlchemyAPI and Node-RED

by Armen Pischdotchian – World of Watson 2016

Download the following Lab Guide :

http://bluemix-watson-day.mybluemix.net/files/Lab4-Face_Recognition.pdf

Before Starting:

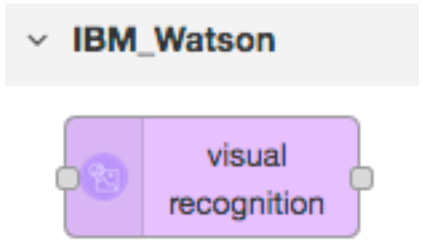
If you successfully executed Lab 3 with Node-RED and if your Node-RED app is still alive, keep it: you have the possibility to start the lab at step 7 , page 6/8 , **Populate the Node-RED canvas.** Create a **AlchemyAPI (Visual Recognition)** service bound to your existing Node.js application.

If you did not create a Node-RED application in Lab 3, ignore this message & proceed with the lab.

At the End, Additional step:
Add debug nodes before & after the Watson API calls (nodes): display the input & output objects

Last minute update

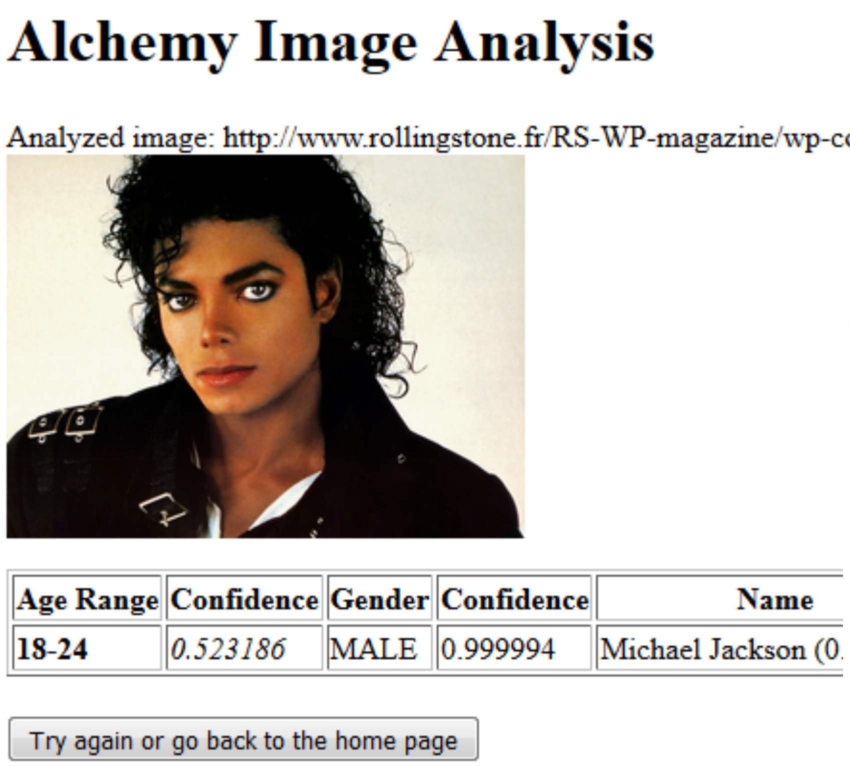
- AlchemyAPI service was upgraded to Visual Recognition Service.
- New Node-red node is



- Page 8, step 33, in node function “Report Faces”, use this HTLM code instead →

```
<h1>Alchemy Image Analysis</h1>
<!-- parse all images -->
{{#payload.images}}
<p>Analyzed image: {{source_url}}<br/><img id="alchemy_image"
src="{{source_url}}" height="200"/></p>
{{^faces}}
<P>No Face detected</P>
{{/faces}}
<table border='1'>
<thead><tr><th>Age Range</th>
<th>Confidence</th>
<th>Gender</th>
<th>Confidence</th>
<th>Name</th>
</tr>
</thead>
<!-- parse all faces -->
{{#faces}}
<tr>
<td><b>{{age.min}} - {{age.max}}</b></td>
<td><i>{{age.score}}</i></td>
<td>{{gender.gender}}</td>
<td>{{gender.score}}</td>
{{#identity}}
<td>{{identity.name}} ({{identity.score}})</td>{{/identity}}</tr>
{{/faces}}
</table>
{{/payload.images}}<form action="{{req._parsedUrl.pathname}}"><br>
<input type="submit" value="Try again or go back to the home page"/>
</form>
```

Lab 4 – Expected Result with Watson AlchemyAPI

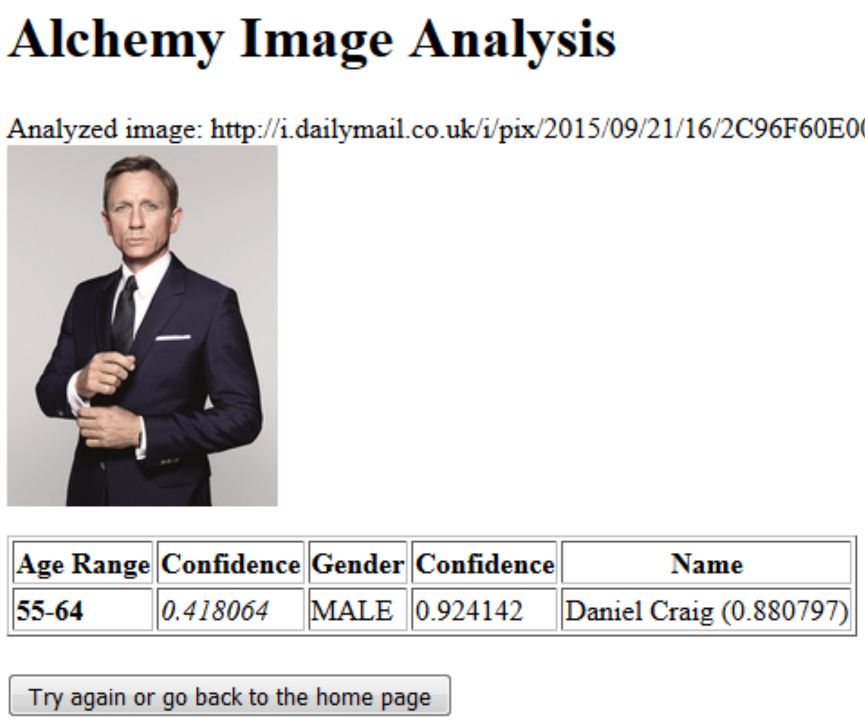


Alchemy Image Analysis

Analyzed image: http://www.rollingstone.fr/RS-WP-magazine/wp-content/uploads/2015/06/michael1.jpg

Age Range	Confidence	Gender	Confidence	Name
18-24	*0.523186*	MALE	0.999994	Michael Jackson (0.

Try again or go back to the home page

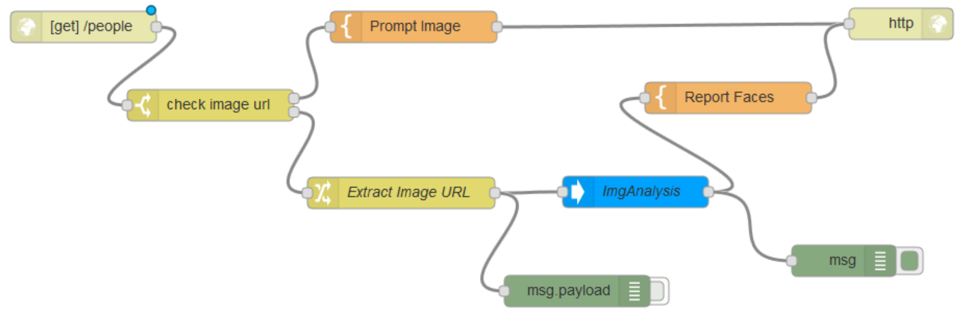


Alchemy Image Analysis

Analyzed image: http://i.dailymail.co.uk/i/pix/2015/09/21/16/2C96F60E00000578-0-image-a-126_1442848442984.jpg

Age Range	Confidence	Gender	Confidence	Name
55-64	*0.418064*	MALE	0.924142	Daniel Craig (0.880797)

Try again or go back to the home page



Lab 5 – IBM Watson Conversation

IBM Watson – Natural Language & Conversation – Weather API

#bluemix

Lab 5 – Instructions

Developing a Chatbot Using the IBM Watson Conversation Service

by Armen Pischdotchian – World of Watson 2016

Refer to the following Lab Guide :

https://bluemix-watson-day.mybluemix.net/files/Lab5-conversation_expedited_v4.pdf

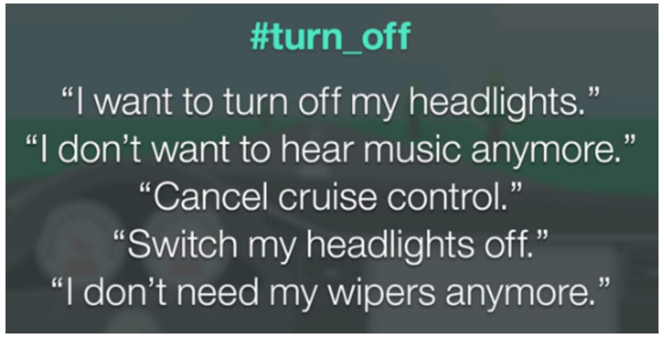
Download Lab Materials

https://bluemix-watson-day.mybluemix.net/files/Lab5_ConversationMaster.zip

You will deploy your application on your local desktop, on a node.js runtime. At the end of the lab, you have the possibility to deploy it in Bluemix.

Lab 5 – Watson Discussion: A few notions

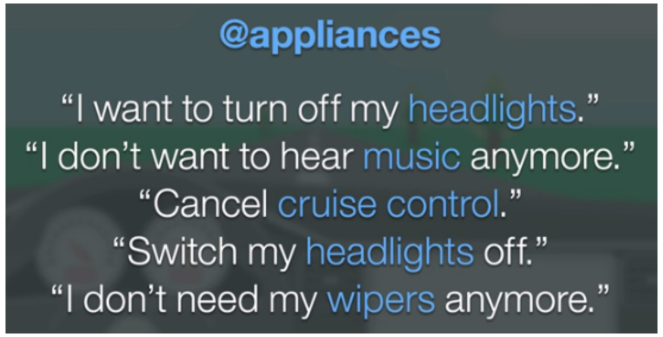
1. Import or create **Intents** & Examples



#turn_off

"I want to turn off my headlights."
"I don't want to hear music anymore."
"Cancel cruise control."
"Switch my headlights off."
"I don't need my wipers anymore."

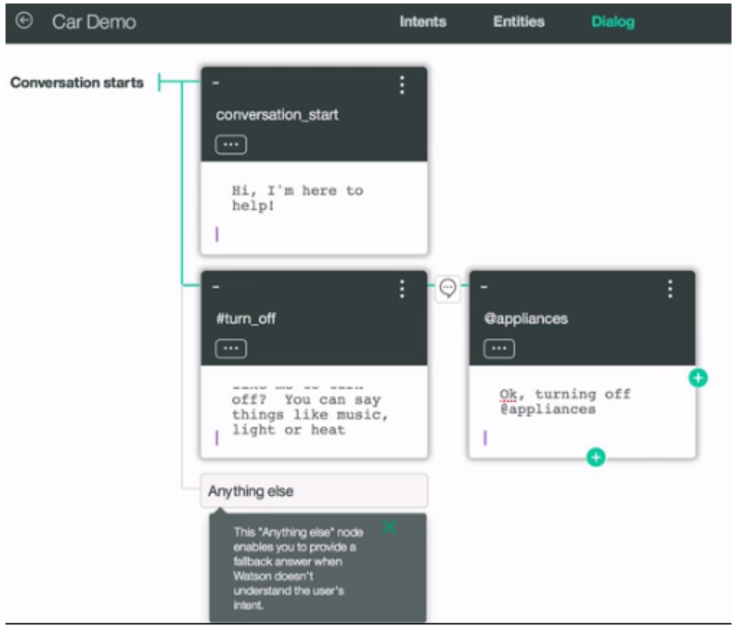
2. Import or create **Entities** & Values



@appliances

"I want to turn off my headlights."
"I don't want to hear music anymore."
"Cancel cruise control."
"Switch my headlights off."
"I don't need my wipers anymore."

3. Import or Configure **Dialog**



conversation_start

Hi, I'm here to help!

#turn_off

off? You can say things like music, light or heat

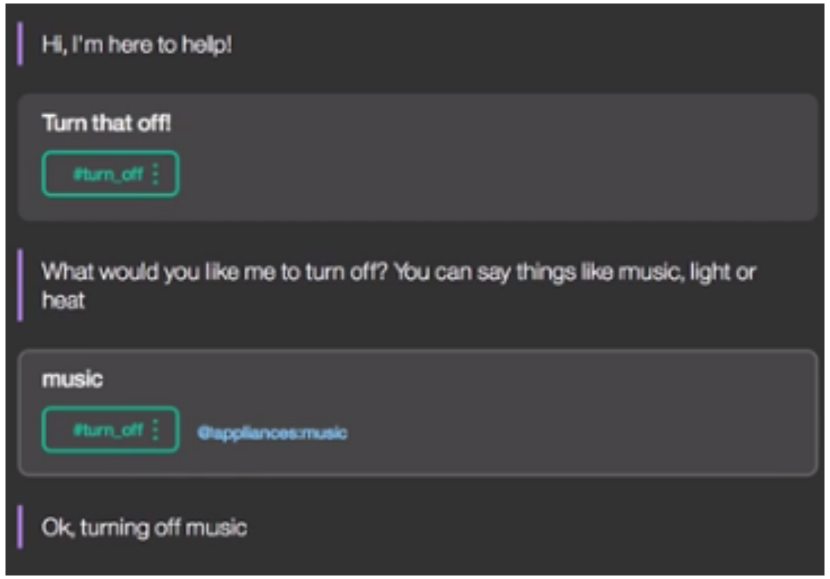
@appliances

Ok, turning off @appliances

Anything else

This "Anything else" node enables you to provide a fallback answer when Watson doesn't understand the user's intent.

4. Adjust & test Dialog



Hi, I'm here to help!

Turn that off!

#turn_off

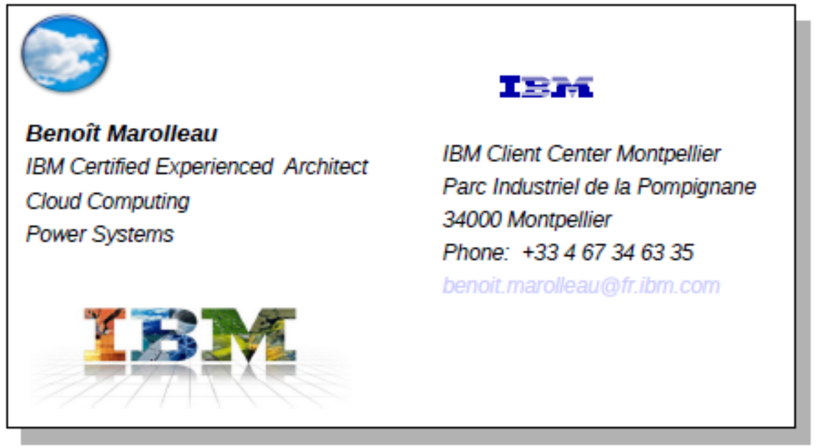
What would you like me to turn off? You can say things like music, light or heat

music

#turn_off @appliances:music

Ok, turning off music

End of Lab



Benoît Marolleau
IBM Certified Experienced Architect
Cloud Computing
Power Systems

IBM Client Center Montpellier
Parc Industriel de la Pompignane
34000 Montpellier
Phone: +33 4 67 34 63 35
benoit.marolleau@fr.ibm.com

Optional Labs

To go further…

#bluemix

Lab 6 – Train Custom Images

IBM Watson – Visual Recognition APIs: /classify /detect_faces /recognize_text

#bluemix

Lab 6 – Visual Recognition Cheat Sheet

You'll find the complete Lab Guide from Armen Pischdotchian on github or here.

1. Create a Visual Recognition Service & write down the Service Credentials (API Key)

Visual Recognition
Understand the contents of images. Create custom classifiers to develop smart applications.

2. Install cURL on your desktop.
3. Download useful files from here (or bring your own), unzip, and open a terminal in that folder.
4. Classify Exercise with Fruitbowl.jpg (do the same with prez.jpg, sign.jpg)

```
curl -X POST -F "images_file=@fruitbowl.jpg" https://gateway-a.watsonplatform.net/visual-recognition/api/v3/classify?api_key=<API-KEY>&version=2016-05-20
```

5. Detect Faces with prez.jpg

```
curl -X POST -F "images_file=@prez.jpg" https://gateway-a.watsonplatform.net/visual-recognition/api/v3/detect_faces?api_key=<API-KEY> &version=2016-05-20
```

6. Recognize Text with sign.jpg

```
curl -X POST -F "images_file=@sign.jpg" https://gateway-a.watsonplatform.net/visual-recognition/api/v3/recognize_text?api_key=<API-KEY>&version=2016-05-20
```

7. Create Classifier: Dogs + Positive Examples Classes (Husky, Beagles, Golden Retrievers) and negative examples (Cats).

```
curl -X POST -F "beagle_positive_examples=@Beagle.zip" -F "husky_positive_examples=@Husky.zip" -F "goldenretriever_positive_examples=@GoldenRetriever.zip" -F
"negative_examples=@Cats.zip" -F "name=dogs" https://gateway-a.watsonplatform.net/visual-recognition/api/v3/classifiers?api_key=<API-KEY>&version=2016-05-20
```

8. Test your custom Classifier by Classifying a picture

```
curl -X POST -F "images_file=@dogs.jpg" -F "parameters=@myparams.json" "https://gateway-a.watsonplatform.net/visual-recognition/api/v3/classify?api_key=<API-KEY> &version=2016-05-20"
```

Lab 7 – Building Advanced Dialog in Watson Conversation Service

IBM Watson – Conversation & Dialog

IBM

#bluemix

Lab 7 – Instructions

Building Advanced Dialog in Watson Conversation Service

by Armen Pischdotchian

- ❑ This Lab is based on Lab 5 – "Car bot" , requires you to create a new Conversation Service or workspace.
- ❑ Learn how to build a Dialog with Conversation!
- ❑ Refer to the following Lab Guide:

  https://bluemix-watson-day.mybluemix.net/files/Lab7-Conversation_dialog.pdf

To go Further…

- Reference
  - https://github.com/apischdo/WOW2016
    - IBM Watson Visual Recognition APIs
    - Advanced dialog with Conversation….
    - More to come…
- Watson Developer Cloud (Docs, Demos, Tutorials…)
  - http://www.ibm.com/watson/developercloud/