Lab – Build a connected car IoT app with Geospatial Analytics

Hands-On Lab
# Table of contents

1. Create the Connected Cars Bluemix Application ................................................................. 3
Introduction

In this lab you’ll be building a Bluemix application using IoT foundation and Geospatial service. This tutorial explains how to deploy and extend an Internet of Things (IoT) Connected Car starter kit on Bluemix with the Internet of Things and Geospatial services. The starter kit allows you to simulate, view and manage vehicles driving through a city and set up geofences for notification.

Most IoT applications consist of three pieces:

1) A connected “thing”
2) An application to view and manage the “thing”
3) Analytics to detect events triggered by the “thing”

The Internet of Things service on Bluemix (IoT Foundation) makes it trivially easy to connect a “thing” to applications and analytics services. To demonstrate this, we built an IoT “Starter Kit” for Connected Car. The kit consists of three pieces:

1) **The Thing**: A vehicle simulator (Node.js)
2) HTML5 applications to view and manage vehicles on a map
3) The **Geospatial Analytics** service on Bluemix, and **Node-RED** for analytics

See it in action: http://trafficsimulator.mybluemix.net

This article guides you on how to configure and deploy the “Starter Kit” on Bluemix, and how to build analytics with the Geospatial Analytics service and Node-RED.

What you'll need to build your app

List prerequisites here. For example:

- A Bluemix account and a DevOps Services account, both linked to your IBM ID.
- The Cloud Foundry cf command line tool.
- A basic understanding of Javascript.

Provide the following here:

- The URL to your running app: http://trafficsimulator.mybluemix.net
- The URL to get your code in DevOps Services: https://hub.jazz.net/project/bryancboyd/iot-connected-vehicle-starter-kit/overview
Overview of the Connected Vehicle starter kit

The Connected Vehicle application you will configure and deploy has three components:

1) Node.js vehicle simulator
2) HTML5 Map app
3) HTML5 Tester app

The three applications use IBM IoT Foundation for real-time publish/subscribe messaging with the MQTT protocol. The simulated vehicles publish telemetry data frequently (2 msgs/sec) and subscribe to a command topic whereby they accept commands from the tester application (ex. “set speed to 60”). The Map app subscribes to the vehicle telemetry topic in order to display position and status in real-time. The Tester app allows you to publish control commands to the vehicles and notifications to the Map app.
Step 1. Create the Internet of Things service

1. In your browser go to Bluemix URL http://bluemix.net and login if necessary.

2. Make sure you’re in the Dashboard tab (if not click on the Dashboard link at the top of the page to take you there).

3. In the dashboard, select ADD A SERVICE. Choose Internet of Things from the service catalog.
Choose the selected plan of **Free** and select **CREATE** to create the service.

4. Select the newly-created service in your dashboard, then click the **LAUNCH** button in the top-right corner of the page.

5. You are now in the Internet of Things Foundation (IoTF) dashboard, where you can register devices, generate API keys for applications, view your usage, and invite others to the organization.
Lab : Connected cars

Step 2. Configure the IoT Foundation Organization

The Connected Vehicle application uses IoT Foundation for near real-time messaging between simulated vehicles and the Map and Tester apps. To facilitate this communication, you first have to register the devices and generate an API key for the applications.

The vehicle simulator allows you to model multiple vehicles from a single Node.js runtime instance. Each vehicle simulator will be treated as a device by IoT Foundation. You will eventually run 3 vehicle simulators, so the first step is to manually register these simulators to obtain access credentials.

1. From the organization dashboard's Devices tab, select Add Device. In the Device Type dropdown, select Create a device type... and enter vehicle (in lower case) in the field below. Choose a device ID of any length (alphanumeric characters only) that will be unique to this organization (ex. ABC, DEF, GHI) and select Continue.

   ![Device Registration](image)

   2. The next screen shows you the credentials for your device. **Open a text editor and write these down!** For example:

```
org=o4ze1w
type=vehicle
id=ABC
auth-method=token
auth-token=5QK1rWHG9JhDw-rs+S
```

   3. Create two more devices (ex. DEF, GHI), and write down the credentials. You will configure the starter kit with this information.
4. From the organization dashboard, click on the API Keys tab and select **New API Key**.

Copy down the key and token. The Map and Tester app will use these credentials to connect to IoT Foundation.
Step 3. Download, configure, and deploy the starter kit

The source code for IoT starter kit is stored in an IBM DevOps Service project, so there are multiple ways that you can obtain your own copy of the code, including either of the following:

- Creating your own project using the project using the FORK PROJECT option on the project page.
- Cloning the git repository in your local machine and using Cloud Foundry command line interface.

In this lab exercise, you’ll use FORK PROJECT option to create your own project.

1. Open another tab in your browser, and login to Bluemix Devops services at [http://hub.jazz.net](http://hub.jazz.net). Click on Explore and search for iot-vehicle.

2. Search results will be displayed on Devops dashboard. Click on iot-vehicle-geospatial-starter-kit project. Once the project is opened in your browser, click on Fork Project.
3. Give your project a new name. Please make sure you check the **Make this a Bluemix Project** checkbox. Click on **Create**.

4. You just created your own project. Click on **Edit Code**. Open **manifest.yml**. Pick a globally unique host name (ex. **bryancoyd-trafficssim**) and enter this name for the **host** field. Set the number of instances to **3**, then save the file.
5. Open `config/settings.js`. This file stores all device and API key configuration data.
   1. For `iot_deviceType`, enter `vehicle`
   2. For `iot_deviceOrg`, enter your 6-character organization ID (ex. O4ze1w)
   3. For `iot_deviceSet`, enter the ID and tokens for the three devices you registered
   4. For `iot_apiKey`, enter the API key you created
   5. For `iot_apiToken`, enter the API key token
   6. Keep the default values for `notifyTopic` and `inputTopic`: these are configuration parameters used by the Geospatial Analytics service.
   7. Save your changes.
6. Click on the Deploy link to deploy the application.
7. After your application deploys, run the Map app at http://<app-name>.mybluemix.net. You will see three simulated vehicles moving across the map. Click on a vehicle to view telemetry data.

8. Each vehicle simulator can simulate multiple vehicles. The count is controlled by a Bluemix environment variable: VEHICLE_COUNT.
Step 4. Create and configure Geospatial Analytics

The Bluemix Geospatial Analytics service allows you to track when each vehicle enters or leaves a defined geofence region. For an introduction to the service, see [this article](note: Link to Mike Branson's article). The Map app can interact with the service, allowing you to graphically create and delete geofences dynamically.

1. In the Bluemix dashboard for your new application, select **ADD A SERVICE**. Choose **Geospatial Analytics** from the service catalog. Choose the selected plan of **Free** and select **CREATE** to bind the service to your application.

2. The starter kit application includes APIs to start and stop the Geospatial Analytics with the configured IoT Foundation organization and messaging credentials. To start the service, visit the following URL in your browser and wait for the page to show success (may take 30-45 seconds):
   http://<app-name>.mybluemix.net/GeospatialService_start

3. Open the Map app and create some geofences. Use the Alert button on the toolbar to start the geofence creation process, and drag the handles and edges to position the geofence.

4. To create the geofence, click on the center black circle to open the context menu then select **Create**.
The Geospatial Analytics service will add this geofence to the list of watched regions, and will publish an MQTT message through IoT Foundation whenever a vehicle enters or leaves the area. The Map app will consume the message and display an overlay over the vehicle on the map. To delete a geofence, select the region and press the **Delete** button.
Step 5. Use the Tester app

The Tester app is used to send commands to the simulated vehicles and the Map app. The vehicle simulator subscribes to commands of type `setProperty` and will dynamically change its own properties, speed, and state. The Map app subscribes to commands of type `addOverlay` and will dynamically display a popup of text over a vehicle.

1. Open the Map app and Tester app (http://<app-name>.mybluemix.net/tester) side-by-side, so that you can see both windows.

2. `setProperty` – Select a vehicle and enter the ID in the bottom form on the Tester app. Enter a property of `speed` and a value of 100 and press **Update Property**. You will see the MQTT topic and payload containing the command in the Tester app, and the selected vehicle will change speed to 100 mph.

   • NOTE: the vehicles simulate a set of **static properties** (location, speed, state, and type) and **custom properties**. The `setProperty` API allows you to dynamically add/change/delete a custom property on a vehicle. To add a property, publish a property that doesn’t yet exist (ex. DriverWindow = UP). To delete a property, update the property with a value of an empty string: the vehicle will stop including the property in its telemetry message.
3. **addOverlay** – In the Tester app, use the upper form to display a message over a vehicle.
**Step 6. Create a Node-RED analytics application**

You can use the Node-RED boilerplate in Bluemix to extend the starter kit with basic analytics. Node-RED is a visual tool for wiring together events in IoT use cases. Node-RED allows you to design flows of logic that link a variety of inputs (ex. MQTT, TCP, HTTP) to outputs (ex. MQTT, HTTP, Twitter, MongoDB, Cloudant) to quickly add lightweight analytics.

1. Deploy the Node-RED boilerplate application in Bluemix. In your Bluemix dashboard, click on Catalog, choose the Internet of Things Foundation boilerplate. Provide a name and hostname (ex. bryancoy-nodered) and press Create.

2. When the application is running, select **Bind a Service** and chose the Internet of Things service you created previously.
3. After the application stages, open the Node-RED canvas at the following URL: http://<node-red-app>.mybluemix.net/red/

4. The IoT boilerplate Node-RED application starts with a default flow. Select the nodes on the canvas and delete them.

5. First, build a flow to monitor telemetry for all vehicles. Drag an `ibmiot` input node from the palette on the left onto the canvas. Drag a `debug` node to the right of the `ibmiot` node and wire the two nodes together. Double-click the `ibmiot` node and configure the node like in the image below.

6. Now press **Deploy** (top-right corner of page) and select the **debug** pane. If configuration was successful and the starter kit application is currently running, you will see the MQTT payloads for the connected vehicles in the debug pane.
7. If your VEHICLE_COUNT environment variable is set to a value greater than 1, the telemetry data for VEHICLE_COUNT vehicles will be contained in a single array. To modify the flow to filter for a single vehicle (ex. ABC-2), add a function node in the middle of your flow, double-click and fill in the following code:

```javascript
var data = msg.payload;
for (var i in data) {
    if (data[i].id == "ABC-2") {
        msg.payload = data[i];
        return msg;
    }
}
```

8. Wire the flow together and press Deploy. You should now see data for a single vehicle in your debug pane:
Lab: Connected cars

```
var data = msg.payload;
for (var i in data) {
    if (data[i].id == "ABC-2") {
        msg.payload = data[i];
        return msg;
    }
}
```

See the Info tab for help writing functions.
Step 7. Create a “keyfob” in Node-RED

Next, use Node-RED to build a simple remote control to lock/unlock a vehicle. You will use two inject nodes: one to unlock the vehicle, the other to lock it. For each operation, you will publish a setProperty command for property lockState with a value of "locked" or "unlocked", then publish an addOverlay command to display a notification on the Map app.

1. Go to your Node-RED flow and drag two inject input nodes onto the Node-RED canvas. Edit each node and configure a string payload: "locked" for one, "unlocked" for the other. This payload will be the value that is sent in the setProperty command.

2. Next, add a function node to the flow and wire both inject nodes to it. This function will build and publish two commands to IoT Foundation: one for setting the vehicle state, the other for displaying an overlay message on the Map app. Use the following code:

```javascript
var commandValue = msg.payload;
var setPropertyMsg = {
    deviceId: "ABC",
    deviceType: "vehicle",
    eventOrCommandType: "setProperty",
    format: "json",
    payload: JSON.stringify({
        "id": "ABC-2",
        "property": "lockState",
        "value": commandValue
    })
};
var addOverlayMsg = {
    deviceId: "tester",
    deviceType: "api",
    eventOrCommandType: "addOverlay",
    format: "json",
    payload: JSON.stringify({
        "id": "ABC-2",
        "text": commandValue,
        "duration": 5000
    })
};
return [ [setPropertyMsg, addOverlayMsg] ];
```
3. Next, add an ibmiot output node to the flow, after the function node. The function node will overwrite many of the configuration values on the ibmiot node; these fields we will mark with a "dummy" value of **override**:

![ibmiot output node configuration](image)

4. Now deploy the application (click **Deploy** in the top-right corner of the page) and press the button on the inject nodes while watching the Map app. You will see the lockState property update for vehicle ABC-2, and a message will appear on the map:

![Map app with unlocked car](image)
**Step 8. Tweet geospatial events with Node-RED**

With the Bluemix Geospatial Analytics service, you can create geofence polygons in the Map app and receive IoT Foundation events when a vehicle enters or leaves a region. The Map app already subscribes to this event topic and displays messages as a notification overlay over the vehicle. This geospatial event topic can be subscribed to by multiple applications, and be a building block used to create more complex scenarios. For example: a bounding box for young drivers, signaling parents whenever their child drives outside of an "approved" zone. Or: detect when my vehicle has entered a geofence surrounding my house, and notify my house automation system to prepare for my arrival.

You can use Node-RED to link geofence alerts to other applications and APIs. In this flow, you will publish a Twitter **tweet** when receiving a geofence notification message for a particular vehicle.

1. First, drop an **ibmiot** input node onto the canvas. Configure the node as follows:

   ![Edit ibmiot in node](image)

   ```
   msg.payload = {
       time: msg.payload.time,
       id: msg.payload.deviceInfo.id,
       lon: msg.payload.deviceInfo.location.longitude,
       lat: msg.payload.deviceInfo.location.latitude,
       eventType: msg.payload.eventType,
       region: msg.payload.regionId
   }
   if (msg.payload.id == "ABC-3") { return msg; }
   ```

2. Add a function node after the **ibmiot** input node. This function will filter the data returned from the geospatial service, and only continue flow execution if the vehicle for the alert matches an ID specified. Use the following code (if desired, replace the ID with a vehicle in your environment), wire the function to a debug node, and deploy:

   ```
   msg.payload = {
       time: msg.payload.time,
       id: msg.payload.deviceInfo.id,
       lon: msg.payload.deviceInfo.location.longitude,
       lat: msg.payload.deviceInfo.location.latitude,
       eventType: msg.payload.eventType,
       region: msg.payload.regionId
   }
   if (msg.payload.id == "ABC-3") { return msg; }
   ```

3. Next, drag a **twitter** output node onto the canvas, and wire the output of the function to the input of this node. The twitter node allows you to configure an existing Twitter account using OAuth, then will tweet the contents of **msg.payload**. Sign in to your Twitter account, then modify the function node to build the message for Twitter:

   ```
   msg.payload = {
       time: msg.payload.time,
       id: msg.payload.deviceInfo.id,
   ```
Lab : Connected cars

```javascript
lon: msg.payload.deviceInfo.location.longitude,
lat: msg.payload.deviceInfo.location.latitude,
eventType: msg.payload.eventType,
region: msg.payload.regionId
}
if (msg.payload.id == "ABC-3") {
  var verb = "exited";
  if (msg.payload.eventType == "Entry") { verb = "entered"; }
  msg.payload = "Vehicle " + msg.payload.id + " has " + verb + " region " +
  msg.payload.region + ";"
  return msg;
}
```

4. Deploy the application, and Node-RED will publish a tweet whenever a geofence is crossed by your vehicle:
Conclusion

Building an IoT application with the Internet of Things and Geospatial Analytics services in Bluemix is simple and easy. Try it for yourself: extend this starter kit with other Bluemix services, mobile applications, even physical sensors.