Installation Guide – November 2016

Credit for the original version at Band on the Run - <https://github.com/BandOnTheRun>

This document is revised from that version by David Gristwood and Peter Daukintis, 2016

# Band on the Run Architecture

Overall Band on the Run Architecture

Machine generated alternative text:
An ics 
Web Job 

The [UWP](https://msdn.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp) app connects via Bluetooth, using the [Microsoft Band SDK](http://developer.microsoftband.com/bandSDK), to the band to get real time sensor data.

When the app first connects to a band, it gets a special [Azure IoT Hub](https://azure.microsoft.com/en-us/services/iot-hub/) token from the device web app.

The band sensor telemetry is send via http to the Azure IoT Hub, using the token it obtained earlier.

The [Azure Stream Analytics](https://azure.microsoft.com/en-us/services/stream-analytics/) job processes that telemetry data in real time, and outputs data to

[Azure table storage](https://azure.microsoft.com/en-us/documentation/articles/storage-introduction/), for historical analysis and processing.

The interactive path is trigger when the heart rate goes above a certain threshold, and is written to an [Azure Service Bus](https://azure.microsoft.com/en-gb/services/service-bus/) queue.

An [Azure WebJob](https://azure.microsoft.com/en-us/documentation/articles/web-sites-create-web-jobs/) pulls data off the service bus and pumps it to the tracker web site, via a REST API call.

The tracker web site uses [SignalR](http://www.asp.net/signalr) to send the live heart rate data down to bowsers, to be displayed in real time

# Azure Configuration

This install process is not automated so these are step by step setup instructions for the Azure side.

Going through the steps below will give you a good feel for the project architecture and all the moving parts, so when you come to update or debug the project, it will be much easier.

### Pre-requisites

Visual Studio (at the time of writing, we are using Visual Studio 2015 - <https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx>)

Microsoft Azure SDK (at the time of writing, 2.9, installed via the Web Platform Installer at <https://azure.microsoft.com/en-us/downloads/>)

The Azure Cloud explorer is a useful tool to check the data landing in table storage (<https://azure.microsoft.com/en-gb/documentation/articles/vs-azure-tools-resources-managing-with-cloud-explorer/>)

IoT Hub SDK (useful to manage IoT Hub, especially the Device Explorer - <https://azure.microsoft.com/en-us/documentation/articles/iot-hub-sdks-summary/>)

## Storage Account

There are a number of services in this solution that need storage, including web jobs and stream analytics, so create a storage account, and ensure that the data center region you select will be used throughout the project.

If this is the first resource for the project you have created, create a new Resource Group, and add all future resources in the solution to this resource group

## IoT Hub

Go to the new Azure portal, and create a new IoT Hub

Article on managing IoT hubs through the Azure portal - <https://azure.microsoft.com/en-gb/documentation/articles/iot-hub-manage-through-portal/>

Machine generated alternative text:
IOT hub 
Name 
Name your ,'wb 
* Pricing and scale tier 
Sl - Standard 
* IOT Hub units O 
* partitions O 
4 partitions 
* Resource group 
New resource group name 
* Subscription 
Azure (davidgri@microsoffcom) 
Enable Device Management—PREVIEW O 
Pin to dashboard 
Create 
Choose your pricing and scale tier 
Fl Free 
messages/unit/day 
Device-to-cloud 
Cloud-to-device 
I unit 
0.00 
LED PER IOT HUB UNIT 
Select 
Sl Standard 
400k 
messages/unit/day 
Device-to-cloud 
Cloud-to-device 
200 units maximum 
50.00 
LED PER IOT HUB UNIT 
S2 Standard 
messages/unit/day 
Device-to-cloud 
Cloud-to-device 
200 units maximum 
500.00 
LED PER IOT HUB UNIT 

F1 free is good start for testing.

See <https://blogs.msdn.microsoft.com/david_gristwood/2016/04/12/choosing-the-right-azure-iot-hub-edition/> for more information for scaling IoT hub

Once created, you will need to return back to the "Shared access policies" tab to get and note the primary keys and connection strings to access the IoT hub from elsewhere in the solution

Machine generated alternative text:
iothu bowr,er 
service 
device 
registryRead 
reg ist ryReadWrite 
registry write, service connect, device connect 
service connect 
device connect 
registry read 
registry write 

## Azure Stream Analytics

Create the main streaming analytics job.

Under the Message tab, create a consumer group, which is an "end point" that you will need to hook to later, and use a name such as "streamanalytics".

Machine generated alternative text:
Consumer groups O 
SDefauIt 
p bistreamanalytics 
streamanalytics 

Create the stream analytics Query:

Machine generated alternative text:
Running 
Essentials 
Jab Topology 
Inputs 
1 
input 
Query 
Add tiles 
Outputs 
2 
sOqueueOutput 
TableOutput 

Enter the query:

Machine generated alternative text:
I 
3 
4 
7 
9 
12 
/ * log to table for debug and post event * / 
SELECT Deviceld, Heartrate, EventEnqueuedUtcTime as Time 
INTO TableOutput 
FROM input 
Timestamp by EventEnqueuedUtcTime 
/ * sustained high heart rate gets alert */ 
8 SELECT Deviceld, as 8161NT) 
as HeartRate 
max (Event Enq ueuedUtcT ime) 
as Time 
INTO sbqueueOutput 
FROM input 
Timestamp by EventEnqueuedUtcTime 
GROUP BY Deviceld, 
T umbIingWindow(second, 

Create the stream analytic Input:

Machine generated alternative text:
Inputs 
1 
input 
Query 
Add tiles 
Outputs 
input 
2 
sOqueueOutput 
TableOutput 
SOURCE TYPE 
Stream 
IOT hub 

You will need to provide the connection information for your newly created IoT Hub:

Create the stream analytics Outputs:

TableOutput:

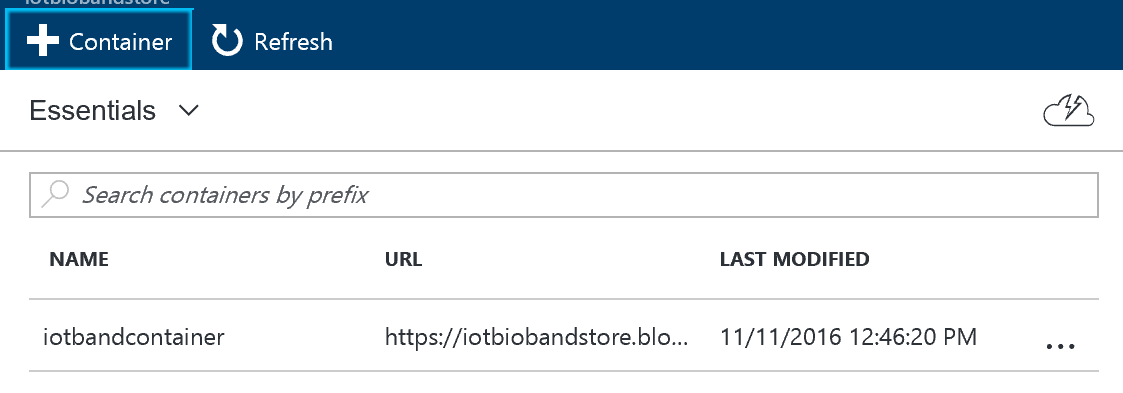
Right now this is outputting to table output and to blob output.

Create a table and a blob on the storage account.

Machine generated alternative text:
Storage account O 
bandontherunstoragenu 
Storage account key 
* Table name O 
BandOnTheRun 
* Partition key O 
Deviceld 
* Row key O 
Time 
* Batch size O 

 blobOutput:

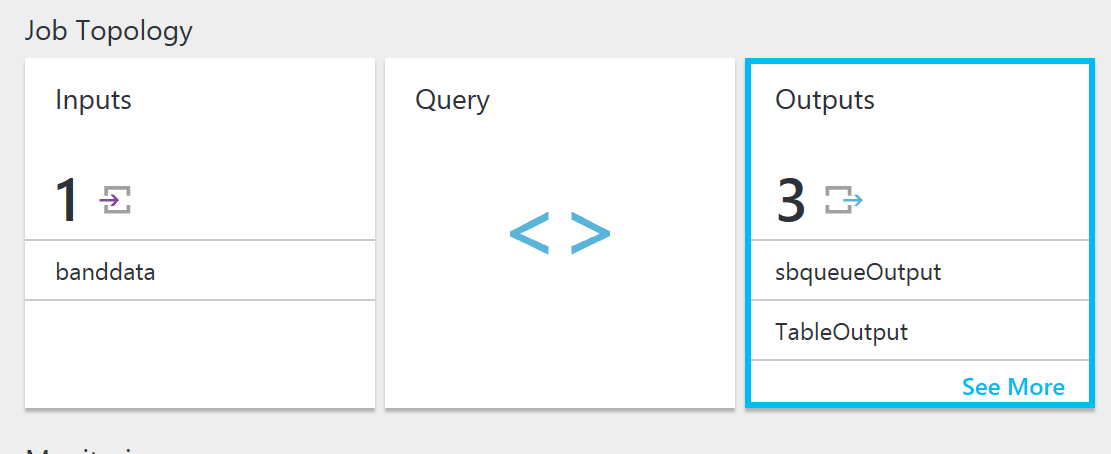
Create a new container for blob storage:



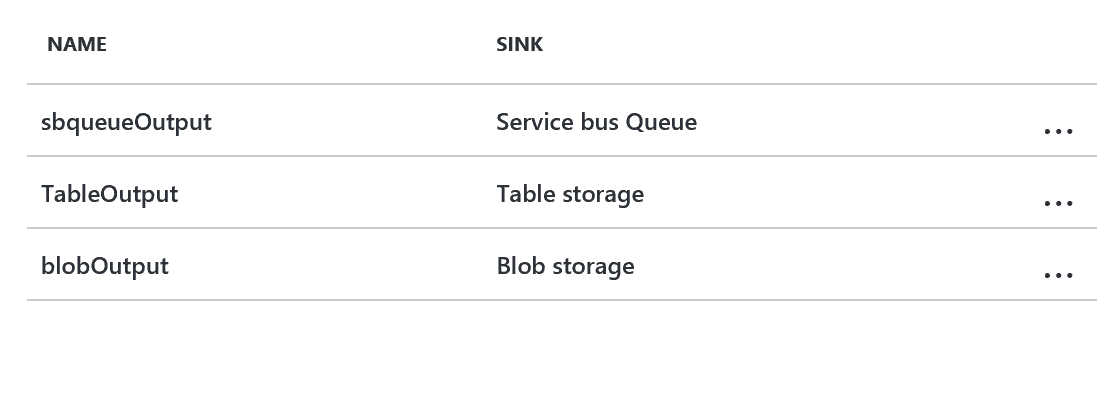
sbqueueOutput:

We output an averaged heartrate data value to the service bus for interactive processing. The idea is to decouple any interactive processing from the core system, so these parts of the system can be evolved and updated without disrupting the message processing

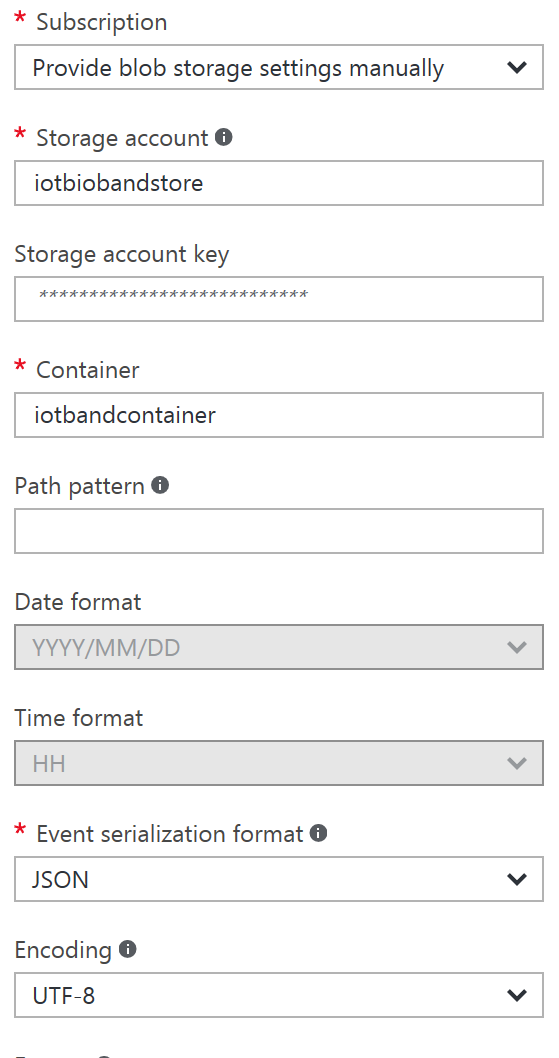
Machine generated alternative text:
Service bus namespace O 
bando ntherunservicebus-ns 
Queue name O 
bandontherunqueue 
* Queue policy name O 
RootManageSharedAccessKey 
Queue policy key O 
* Event serialization formate 
'SON 
Encoding O 
UTF-8 
Format O 
Line separated 



Hook up the Output on this screen using the needed outputs. Here I have table and blob storage:



Hook up container from the blob:



## Service Bus

## 

A service bus is used to link the stream analytics flow and the interactive processing. We deliberately decouple the two, so they can evolve separately. We could use topic/subscription rather than a queue, if multiple listeners are needed.

First, use the old portal (at the time of writing) to create a service bus namespace (and append with "-ns")

Old portal is found at : https://manage.windowsazure.com/

Create namespace:

Machine generated alternative text:
service 
NAMESPACE 
bus 
CREATE A NAMESPACE 
Add a new namespace 
NAMESPACE NAME 
TYPE O 
MESSAGING 
EVENTHUB 
o 
.servicebus.windows.net 
NOTIFICATION HUB 
o 
MESSAGING TIER 
BASIC 
STANDARD 
REGION 
North Euro e 
PREMIUM 

Then create the queue in the new namespace:

Machine generated alternative text:
QUEUE NAME 
QUICK CREATE 
CUSTOM CREATE 
REGION 
North Europe 
NAMESPACE 
bandontherunservicebus-ns 
.servicebus.windows.net 

## Tracking Band Data

You need to be getting Bluetooth input from the band to a running app to get the data loading into Azure.

The app that can collect data is on GitHub:

In future – use the Unity app for this

Output of Stored Data

Web App Data in Progress