Apache Quarks & Raspberry Pi

May 12, 2016

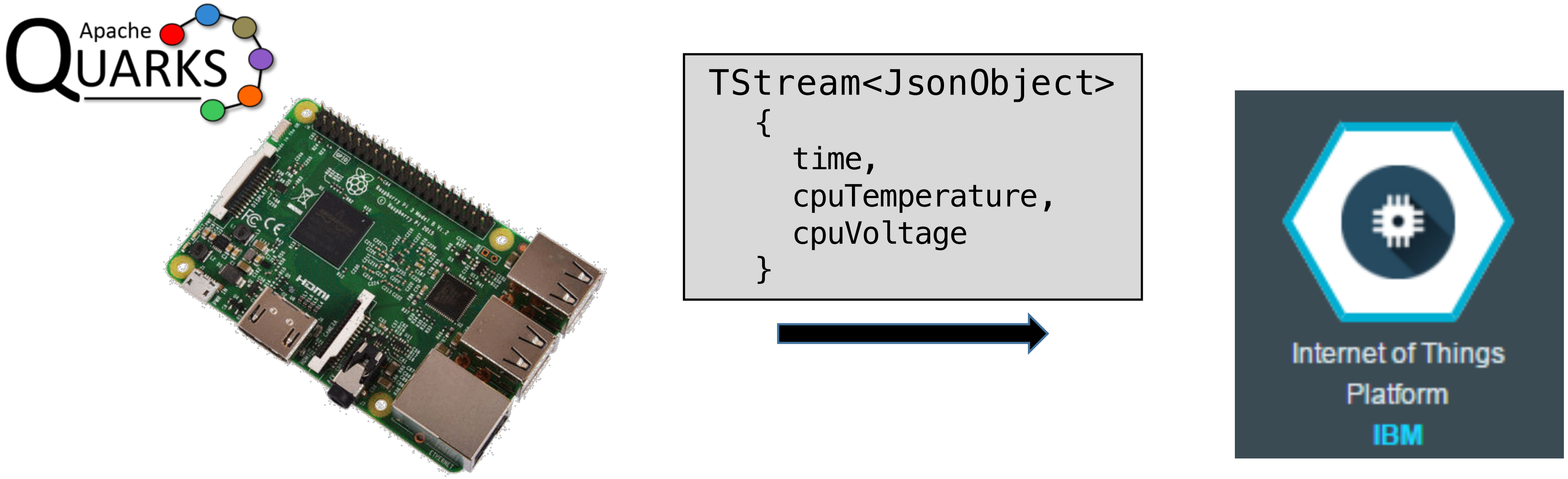
Apache Quarks is currently undergoing Incubation at the Apache Software Foundation.

Topics

- Who am I?
  - Queenie Ma, Apache Quarks contributor and IBM employee

- Register a Raspberry Pi device with IBM Watson IoT Platform
- Set up a Raspberry Pi out of the box with NOOBS/Raspbian OS
- Download Apache Quarks and set up the environment
- Create a Quarks application that uses the Pi4J library to send Pi sensor data (e.g., CPU temperature and voltage) to Watson IoT
- Run the application and view the live data in a browser in Watson IoT

Architecture Diagram



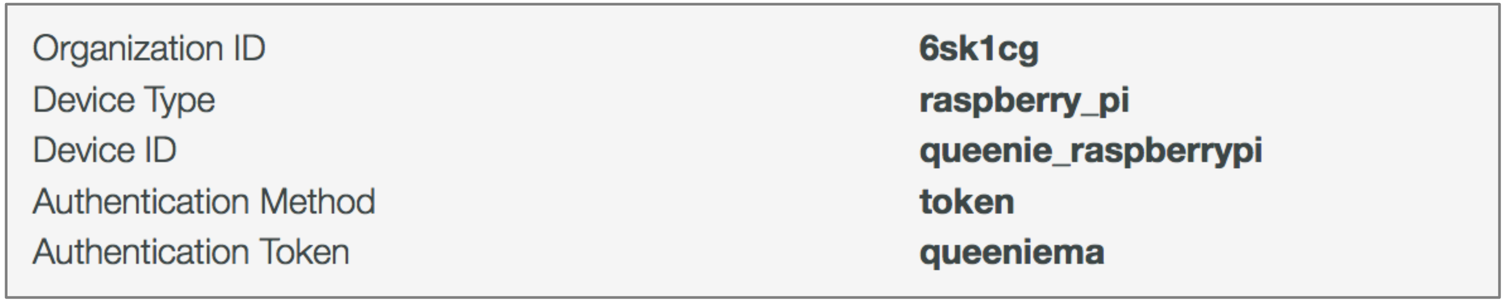
```
TStream<JsonObject>
   {
     time,
     cpuTemperature,
     cpuVoltage
   }
```

Apache Quarks running on a Raspberry Pi device

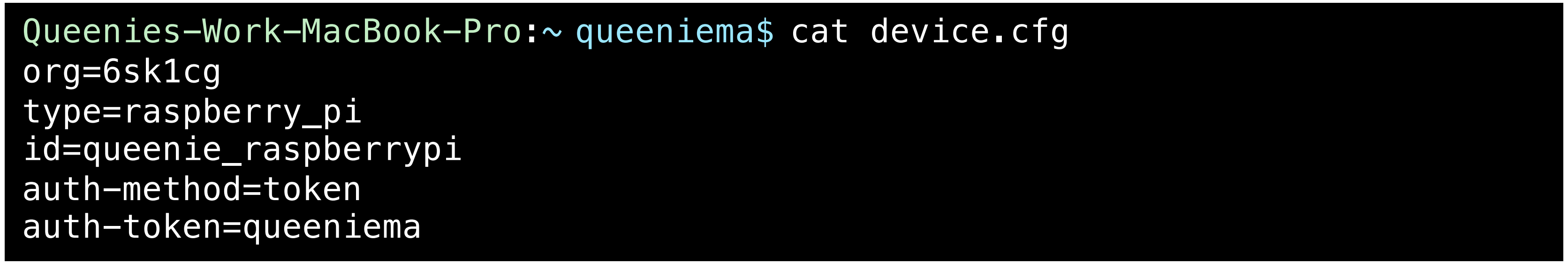
System CPU data information on IBM Watson IoT Platform

Registering with IBM Watson IoT Platform

- Register for a free IBM Bluemix account. Create an IBM Watson IoT Platform Service.
- Launch the Watson IoT Platform dashboard
- Create a device type
- Add a device. Specify the Device ID and provide an authentication token.
- Note the values on the final screen and copy them into a device configuration file



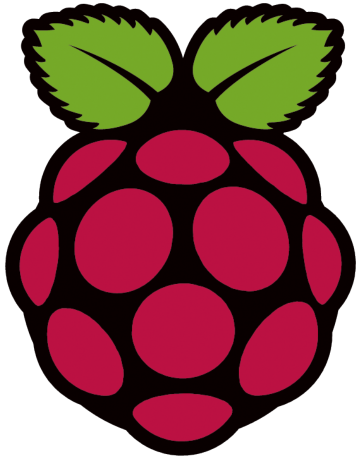
Organization ID	**6sk1cg**
Device Type	**raspberry_pi**
Device ID	**queenie_raspberrypi**
Authentication Method	**token**
Authentication Token	**queeniema**



```
Queenies-Work-MacBook-Pro:~ queeniema$ cat device.cfg
org=6sk1cg
type=raspberry_pi
id=queenie_raspberrypi
auth-method=token
auth-token=queeniema
```

Setting up a Raspberry Pi

- If NOOBS is not already installed on your microSDHC card, install it from the Raspberry Pi Foundation (RPF) website and copy it to the card
- Insert the microSDHC card into the Pi, and plug in the following connections
    - keyboard
    - mouse
    - HDMI cable (to an external monitor)
    - Ethernet cable
- Plug in the power adapter to boot up
- Install an operating system (e.g., Raspbian)



Setting up the Quarks environment

- Prerequisites: Java 8, Apache Ant
- Download the JUnit 4.10 and JaCoCo 0.7.5 JAR files from the internet
  - `junit-4.10.jar`
  - `org.jacoco.agent-0.7.5.201505241946.jar`
  - `org.jacoco.ant-0.7.5.201505241946.jar`
- Copy the JAR files to: `~/.ant/lib`

Quarks application overview

- Prerequisites
  - IBM Bluemix account
  - IBM Watson IoT Platform service created and running
  - Raspberry Pi device registered in Watson IoT Platform
  - Eclipse IDE
- Application flow
  - Retrieve the device configuration file: `device.cfg`
  - Create a `DirectProvider` and `Topology`
  - Connect to IBM Watson IoT Platform via `IotfDevice`
  - Create a *stream* of Raspberry Pi sensor readings, where each tuple contains: the current time, CPU temperature, and CPU voltage. Poll every second.
  - Send the stream of sensor readings as *device events* to IBM Watson IoT Platform
  - Print the readings to `System.out`
  - Submit the topology to the provider

BACKUP SLIDES

Setting up the Quarks environment

- Download the latest Quarks build
  - `wget https://github.com/apache/incubator-quarks/archive/master.zip`
  - Fork the *incubator-quarks* GitHub repository and clone your fork
- In the top-level project directory, run `ant` to build Quarks (builds all code, Javadoc, and JARs into the `./target` directory)
- We can now write and run Quarks applications!

Setting up the Quarks app environment

- Download the Pi4J 1.0 JAR file from the internet
  - `pi4j-core.jar`
- In Eclipse, create a new Java Project and add the necessary JARs to the build path
  - Right-click on the project → *Build Path* → *Configure Build Path…*
  - Click on the *Libraries* tab, then *Add External JARs*
    - `.../target/java8/ext/google-gson-2.2.4/gson-2.2.4.jar`
    - `pi4j-core.jar`
    - `.../target/java8/lib/quarks.api.topology.jar`
    - `.../target/java8/connectors/iot/lib/quarks.connectors.iot.jar`
    - `.../target/java8/connectors/iotf/lib/quarks.connectors.iotf.jar`
    - `.../target/java8/lib/quarks.providers.direct.jar`

Writing the Quarks application

- Create a new Java class
  - Right-click on the project → *New* → *Class*
  - Give the class a name (e.g. `IotfRaspberryPiSensor`) and press the *Finish* button

```
import java.io.File;
import java.util.concurrent.TimeUnit;

import com.google.gson.JsonObject;
import com.pi4j.system.SystemInfo;

import quarks.connectors.iot.IotDevice;
import quarks.connectors.iot.QoS;
import quarks.connectors.iotf.IotfDevice;
import quarks.providers.direct.DirectProvider;
import quarks.topology.TStream;
import quarks.topology.Topology;

public class IotfRaspberryPiSensor {

    public static void main(String[] args) {
```

Writing the Quarks application

```java
// Continued from the previous slide inside of main()...

String deviceCfg = args[0];

DirectProvider dp = new DirectProvider();
Topology topology = dp.newTopology("RaspberryPiSensor");

// Declare a connection to the IBM Watson IoT Platform Bluemix service.
IotDevice device = new IotfDevice(topology, new File(deviceCfg));

// The stream of sensor readings from the Raspberry Pi, where the sensor
// is polled every second.
TStream<JsonObject> readings = systemInfo(topology, 1, TimeUnit.SECONDS);
```

Writing the Quarks application

```
// Continued from the previous slide outside of main()...

public static TStream<JsonObject> systemInfo(Topology topology, long period,
        TimeUnit unit) {
    return topology.poll(() -> {
        JsonObject piInfo = new JsonObject();
        piInfo.addProperty("time", System.currentTimeMillis());

        try {
            piInfo.addProperty("cpuTemperature", SystemInfo.getCpuTemperature());
            piInfo.addProperty("cpuVoltage", SystemInfo.getCpuVoltage());
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        return piInfo;
    }, period, unit);
}
```

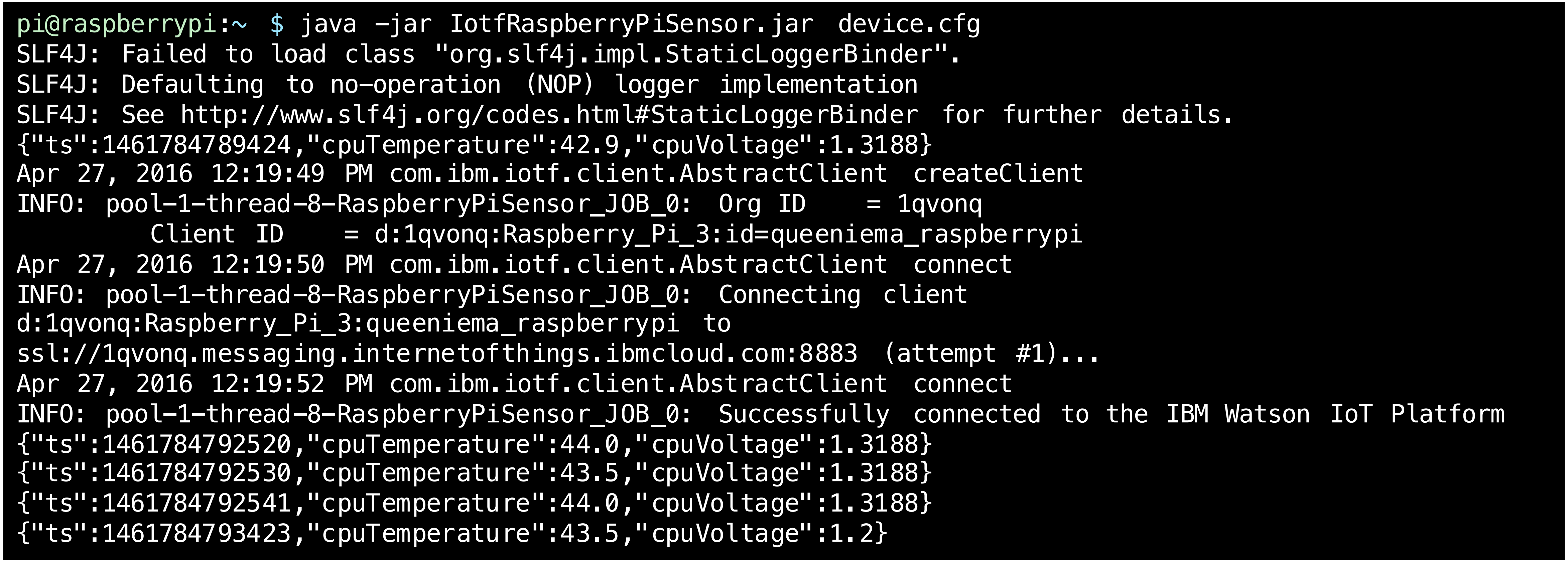
Writing the Quarks application

```
        // Continued from the previous slide inside of main()...

        // Send the device stream as IoT device events to Watson IoT Platform
        // with event identifier "sensor".
        device.events(readings, "sensor", QoS.FIRE_AND_FORGET);

        // Print the readings to the screen.
        readings.print();

        dp.submit(topology);
    }
```

Running the Quarks application

- Export the Java class as runnable JAR file
  - Right-click on the Java file → *Export...*
  - Search for *Runnable JAR file* and select it
  - For *Launch configuration*, select the appropriate Java class
  - For *Export destination*, specify any path
  - For *Library handling*, select *Package required libraries into generated JAR*
  - Press the *Finish* button
- Transfer the generated JAR file and device configuration file to somewhere on the Raspberry Pi (e.g., the home directory)
- Time to run the application!

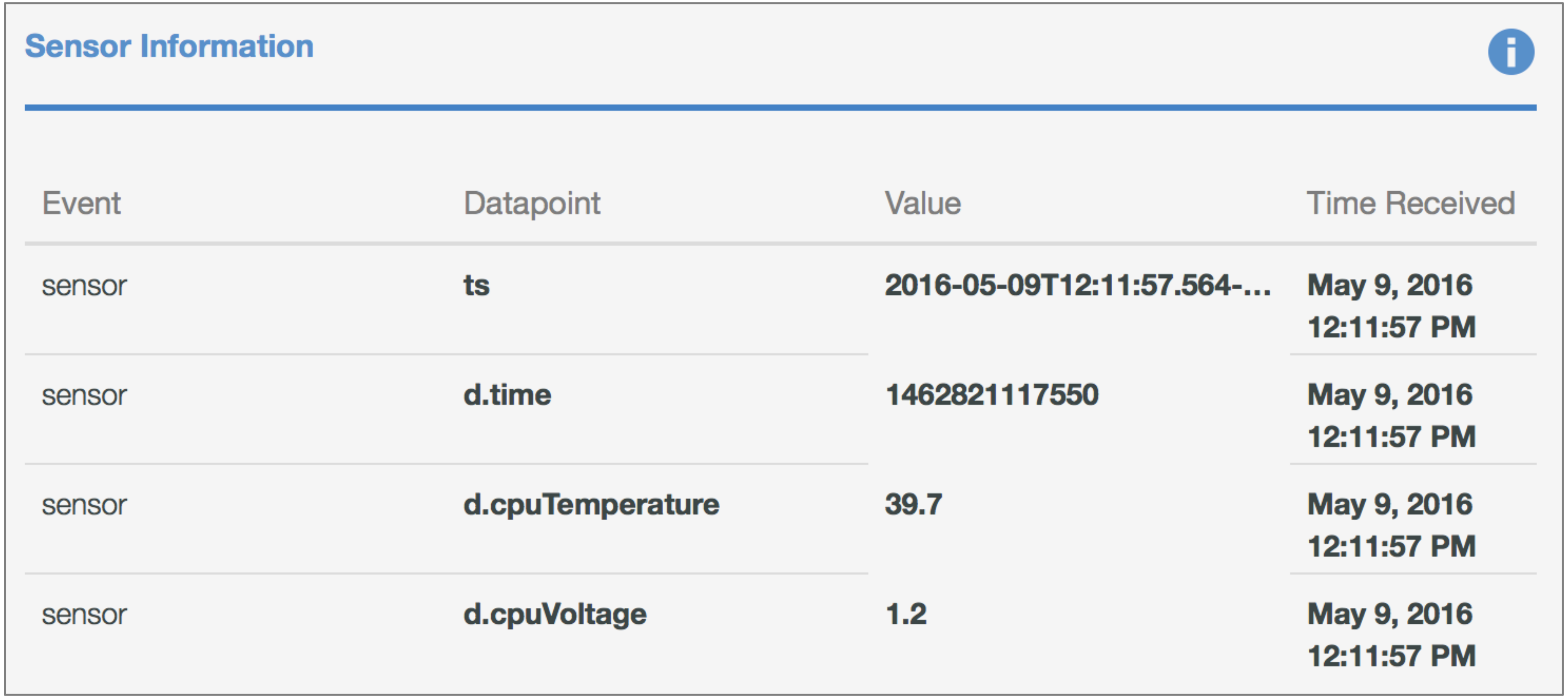
Running the Quarks application



```
pi@raspberrypi:~ $ java -jar IotfRaspberryPiSensor.jar device.cfg
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
{"ts":1461784789424,"cpuTemperature":42.9,"cpuVoltage":1.3188}
Apr 27, 2016 12:19:49 PM com.ibm.iotf.client.AbstractClient createClient
INFO: pool-1-thread-8-RaspberryPiSensor_JOB_0: Org ID    = 1qvonq
        Client ID    = d:1qvonq:Raspberry_Pi_3:id=queeniema_raspberrypi
Apr 27, 2016 12:19:50 PM com.ibm.iotf.client.AbstractClient connect
INFO: pool-1-thread-8-RaspberryPiSensor_JOB_0: Connecting client
d:1qvonq:Raspberry_Pi_3:queeniema_raspberrypi to
ssl://1qvonq.messaging.internetofthings.ibmcloud.com:8883 (attempt #1)...
Apr 27, 2016 12:19:52 PM com.ibm.iotf.client.AbstractClient connect
INFO: pool-1-thread-8-RaspberryPiSensor_JOB_0: Successfully connected to the IBM Watson IoT Platform
{"ts":1461784792520,"cpuTemperature":44.0,"cpuVoltage":1.3188}
{"ts":1461784792530,"cpuTemperature":43.5,"cpuVoltage":1.3188}
{"ts":1461784792541,"cpuTemperature":44.0,"cpuVoltage":1.3188}
{"ts":1461784793423,"cpuTemperature":43.5,"cpuVoltage":1.2}
```

Demo

- View the live sensor data in IBM Watson IoT Platform



Sensor Information

Event	Datapoint	Value	Time Received
sensor	**ts**	**2016-05-09T12:11:57.564-...**	**May 9, 2016 12:11:57 PM**
sensor	**d.time**	**1462821117550**	**May 9, 2016 12:11:57 PM**
sensor	**d.cpuTemperature**	**39.7**	**May 9, 2016 12:11:57 PM**
sensor	**d.cpuVoltage**	**1.2**	**May 9, 2016 12:11:57 PM**

Questions / Reference

- Apache Quarks website
  - http://quarks.incubator.apache.org/
- Apache Quarks: DEVELOPMENT.md, more information about development environment
  - https://github.com/apache/incubator-quarks/blob/master/DEVELOPMENT.md
- Download NOOBS (New Out Of the Box Software), an OS installation manager
  - https://www.raspberrypi.org/downloads/noobs/
- Set up a VNC server for remote access
  - https://www.raspberrypi.org/documentation/remote-access/vnc/
- JUnit 4.10 and JaCoCo 0.7.5 JARs
  - http://search.maven.org/remotecontent?filepath=junit/junit/4.10/junit-4.10.jar
  - http://search.maven.org/remotecontent?filepath=org/jacoco/jacoco/0.7.5.201505241946/jacoco-0.7.5.201505241946.zip
- Pi4J 1.0 JAR
  - http://get.pi4j.com/download/pi4j-1.0.zip
- IBM Bluemix and IBM Watson IoT Platform
  - http://www.ibm.com/cloud-computing/bluemix/
  - https://internetofthings.ibmcloud.com/