To continue from where the last lecture finished you should be in the terminal window in the directory where your server.js, package.json, manifest.ml and .cfignore are located. You also need to be logged onto Bluemix in the cf tool. You can verify that you are logged on to the correct organisation, region and space by issuing the cf t command

To be able to access the Internet of Things service from your application you need to bind your application to the service. To do this you first need to know what the service is called. Run the command

```
cf s
```

You need to copy the name of the iotf-service, as this needs to added to the **manifest.yml** file to bind the application to the service.

Update the **manifest.yml** file, replacing the iotf-service name (Internet of Things Platform-o4 in the example below) with the value shown in the output of the cf s command:

```
---
applications:
- name : bi-nodeserver
  instances : 1
  memory : 256M
services:
- Internet of Things Platform-o4
```

Next the server.js application needs to be updated to

- parse out the VCAP_SERVICES environment variable to retrieve the credentials for the IoTF service.
- connect to the IoTF service
- subscribe for device events
- handle device events when they arrive

The following code has all the modifications needed. Please take time to ensure you understand all the changes, referring back to the API documentation if needed.

```
var express = require('express');
var app = express();
var iotf = require('ibmiotf');
var appConfig;

var serverPort = process.env.VCAP_APP_PORT || 3000;
var serverHost = process.env.VCAP_APP_HOST || 'localhost';

if (process.env.VCAP_SERVICES) {
    var env = JSON.parse(process.env.VCAP_SERVICES);
    appConfig = {
                   'org' : env["iotf-service"][0].credentials.org,
                   'id' : 'bi-nodeserver',
                   'auth-key' : env["iotf-service"][0].credentials.apiKey,
                   'auth-token' : env["iotf-service"][0].credentials.apiToken
                  }
} else {
    appConfig = require('./application.json');
}
```

```

var responseString = 'Hello Coursera';

var appClient = new iotf.IotfApplication(appConfig);
app.get('/', function(req, res) {
    res.send(responseString);
});

var server = app.listen(serverPort, serverHost, function() {
    var host = server.address().address;
    var port = server.address().port;
    console.log('Listening at http://%s:%s', host, port);
    appClient.connect();

    appClient.on('connect', function() {
        appClient.subscribeToDeviceEvents();
    });

    appClient.on('deviceEvent', function(deviceType, deviceId, eventType, format
      , payload) {
        responseString = "Device event at " + new Date().toString() + " from " +
          deviceType +
                                ":" + deviceId + "; event = "+ eventType +", payload =
                                  " + payload;
    });

});
```

You'll notice that the return code for the default endpoint has also been modified to return a string, which is initially set to "Hello Coursera" to match the previous version of the code. However, when a device event is received it updates the return data.

A new package, ibmiotf, is now used by the application, so the dependencies section of the **package.json** file needs to be modified to include the new package. This can be done by issuing the command:

```
npm install ibmiotf --save
```

Push the updated **server.js, package.json** and**manifest.yml** files to Bluemix with the **cf push** command. Once it is running point your browser at the application and if your Raspberry Pi is running you will see the event data. Refresh your browser to see the data being updated as new events arrive.

You will notice that the code also includes a case where the**VCAP_SERVICES** environment variable is not defined, i.e. when you are running locally. Here a file, **application.json**is read for the service authentication details. To use this file you need to create an application API key in the Watson IoT platform console and then create an **application.json** file containing the values. The content should be similar to this, but with the org, id, key and token values you for your IoT service.

```
{
    "org" : "zmex21",
    "id" : "bi-nodeserver",
    "auth-key" : "a-zmex21-abcde12345",
    "auth-token" : "!@£$%abcde12345678"
}
```

You may also want to add the **application.json** file name to the **.cfignore** file as it is only used when running locally, so doesn't need to be sent to Bluemix.

You should now be able to run your application locally, but it is still connecting to the Watson IoT Service.