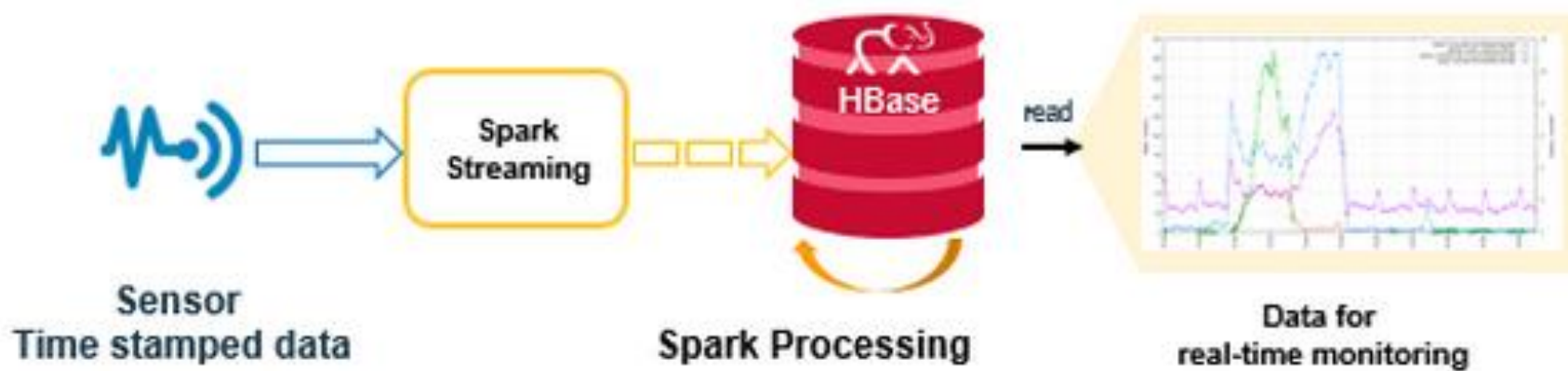


巨量資料實戰課程(III) – 建立即時分析儀表板

David Chiu
2016/12/09

系統架構

如何建立一個即時監控儀表板



建構目標

- 啟動一個Kafka Producer 不斷遞送串流資料
- 啟用一個Spark Streaming 工作，(Consume)消化Producer 遞送過來的Streaming Log，並且將資料存進HBase
- 聚合(Aggregate) Hbase 的資料，並將結果呈現到儀表板上

建立資料API

Flask

■ 什麼是Flask?

a microframework for Python based on Werkzeug, Jinja 2 and good intentions

■ 安裝Flask

pip install flask



第一個Flask APP

```
from flask import Flask  
app = Flask(__name__)
```

hello.py

```
@app.route("/")  
def hello():  
    return "Hello World!"
```

```
if __name__ == "__main__":  
    app.run()
```

打入 python hello.py
執行

預設會在
<http://127.0.0.1:5000/>
看到 Hello World

回傳JSON 資料

```
@app.route("/")  
def api():  
    dt = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')  
    data = random.randint(50,900)  
    return jsonify({'dt':dt, 'data':data})
```


Plotly

安裝Anaconda

Python 3.5 version

64-BIT INSTALLER (351M)

32-BIT INSTALLER (292M)

選擇Python 3.5 版

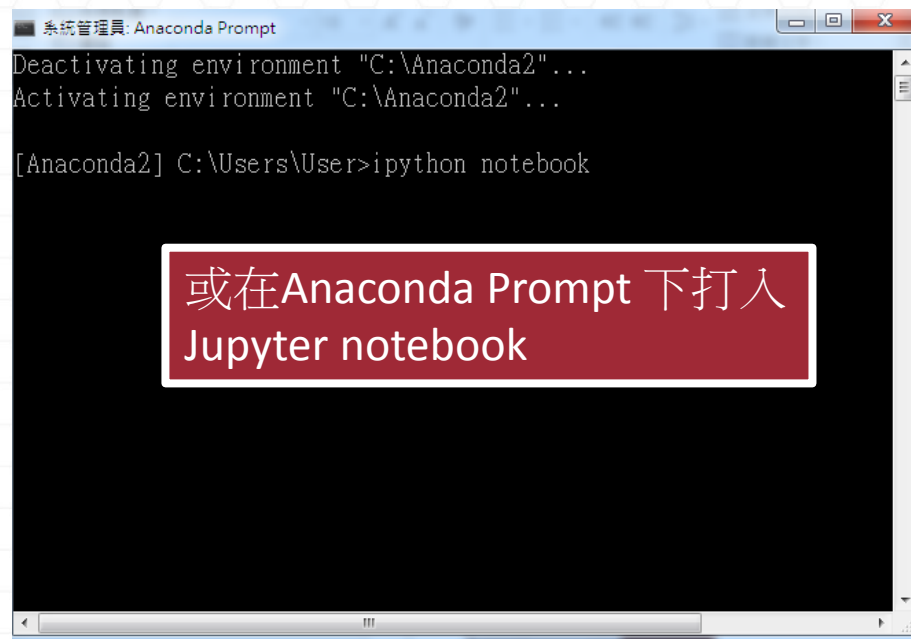
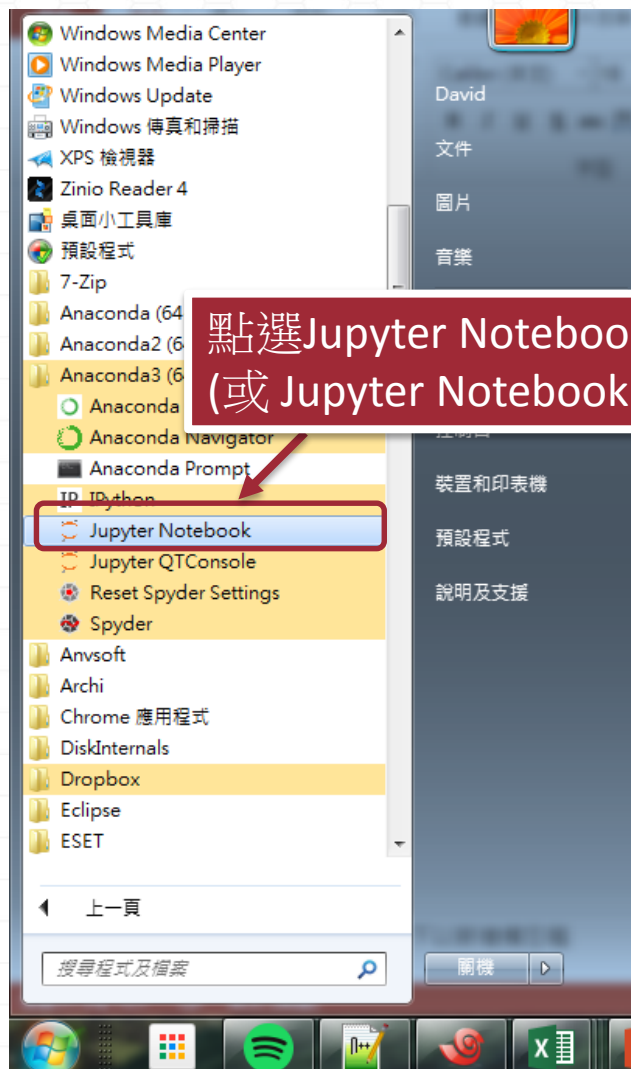
Python 2.7 version

64-BIT INSTALLER (340M)

32-BIT INSTALLER (285M)

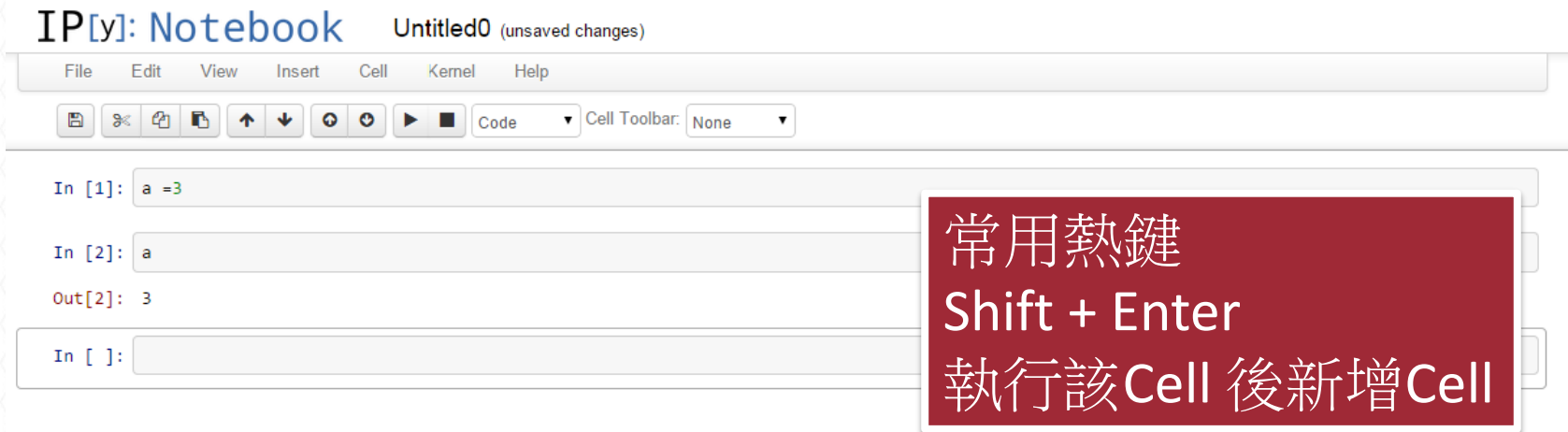
<https://www.continuum.io/downloads>

使用 Jupyter Notebook



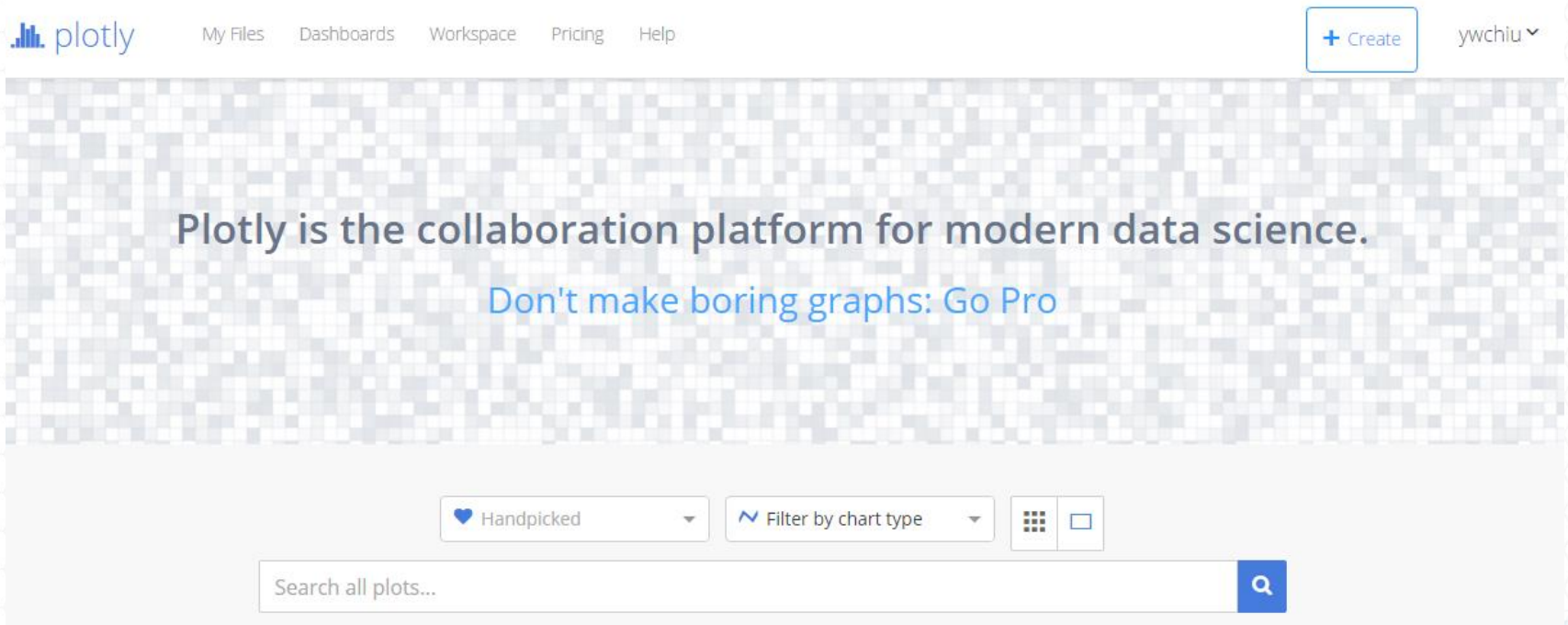
啟用 Jupyter Notebook

- 在命令列下打:
 - Jupyter notebook
 - 自動開啟瀏覽器後便可瀏覽 (預設為localhost:8888)
- 可匯出.ipynb, .py 各種不同格式檔案
- 瀏覽快捷鍵 Help -> Keyboard Shortcuts



Plotly

■ pip install plotly

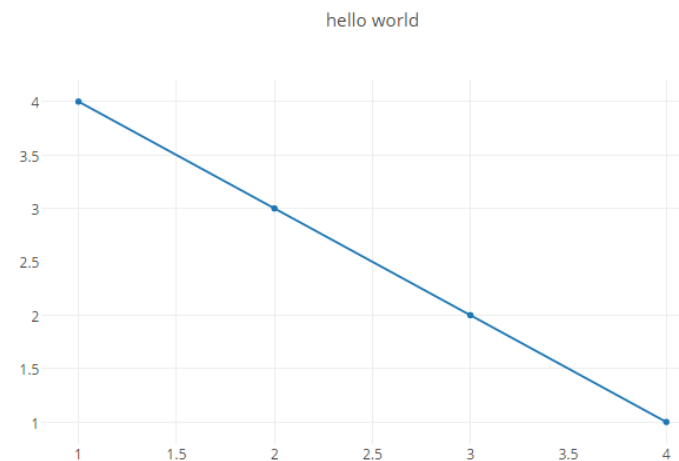


使用Plotly 畫圖

```
import plotly
from plotly.graph_objs import Scatter, Layout

plotly.offline.init_notebook_mode()

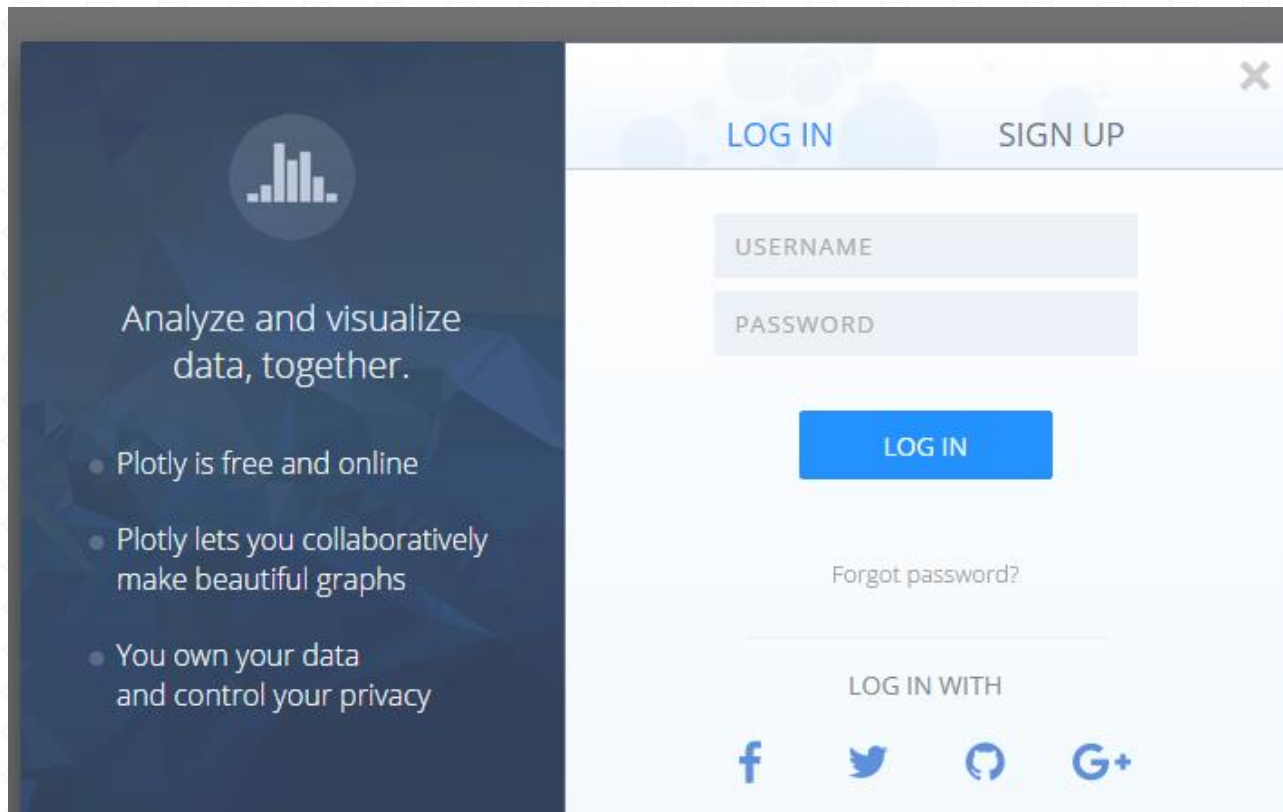
plotly.offline.iplot({
    "data": [Scatter(x=[1, 2, 3, 4], y=[4, 3, 2, 1])],
    "layout": Layout(title="hello world")
})
```



可以Offline 或 Online
繪製圖表

申請API

■ <https://plot.ly/accounts/login/>



The image shows a screenshot of the Plotly login page. On the left, there is a dark blue sidebar with a bar chart icon and the text "Analyze and visualize data, together." Below this, there are three bullet points: "Plotly is free and online", "Plotly lets you collaboratively make beautiful graphs", and "You own your data and control your privacy". On the right, there is a light blue login form with a close button (X) in the top right corner. The form has two tabs: "LOG IN" (selected) and "SIGN UP". Below the tabs, there are two input fields: "USERNAME" and "PASSWORD". A blue "LOG IN" button is positioned below the password field. Below the button, there is a link "Forgot password?". At the bottom, there is a section titled "LOG IN WITH" followed by four social media icons: Facebook, Twitter, GitHub, and Google+.

取得Streaming API Token

The screenshot shows the Plotly web interface. At the top, there's a navigation bar with the Plotly logo, links for 'My Files', 'Dashboards', 'Workspace', 'Pricing', and 'Help', a '+ Create' button, and a user profile 'ywchiu'. On the left, a sidebar lists account settings: 'PROFILE', 'PASSWORD', 'SUBSCRIPTION', 'API KEYS', and 'MAPBOX ACCESS TOKENS'. The 'API KEYS' section is active, displaying a note about API key generation, a masked key, and a 'Regenerate key' button. Below this, the 'STREAMING API TOKENS' section is shown, with a note about using one token per data-stream and a link to documentation. A red box highlights the 'Add a new token' button. At the bottom, there are links to 'getting started' tutorials for Python, MATLAB, or R, and a link to the 'Plotly community site'.

plotly My Files Dashboards Workspace Pricing Help + Create ywchiu

PROFILE
PASSWORD
SUBSCRIPTION
API KEYS
MAPBOX ACCESS TOKENS
View profile

Note that generating a new API key will require changes to your `~/.plotly/.credentials` file.

.....

Regenerate key

STREAMING API TOKENS

Use one streaming token per data-stream. Check out the [documentation](#) to learn more about the Plotly streaming API.

Add a new token

Learn about working with the Plotly API with our "getting started" tutorials for [Python](#), [MATLAB](#), or [R](#).
Still stuck? Chat with developers and data scientists on the [Plotly community site](#).

使用Plotly 繪製 Streaming 圖表

```
import numpy as np
import plotly.plotly as py
import plotly.tools as tls
import plotly.graph_objs as go
```

```
plotly.tools.set_credentials_file(username='ywchiu', api_key='<API>')
```

```
stream_1 = go.Stream(
    token='<Stream Token>', # link stream id to 'token' key
    maxpoints=80           # keep a max of 80 pts on screen
)
```


設定圖表 Layout

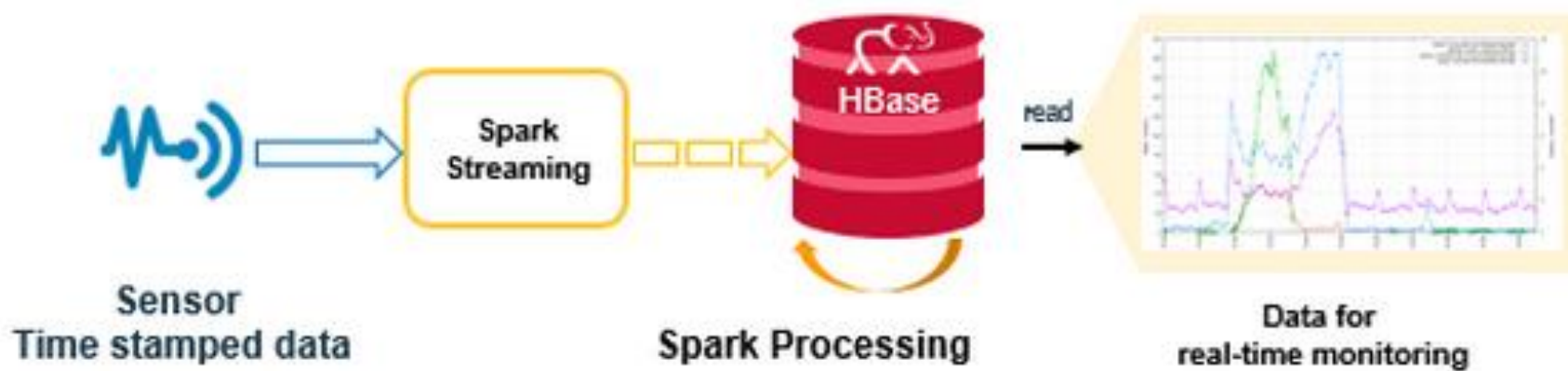
```
trace1 = go.Scatter(  
    x=[],  
    y=[],  
    mode='lines+markers',  
    stream=stream_1      # (!) embed stream id, 1 per trace  
)  
data = go.Data([trace1])  
# Add title to layout object  
layout = go.Layout(title='Time Series')  
# Make a figure object  
fig = go.Figure(data=data, layout=layout)  
# Send fig to Plotly, initialize streaming plot, open new tab  
py.iplot(fig, filename='python-streaming')  
s = py.Stream('w6zijhj2sa')  
s.open()
```

繪製圖表

```
import datetime
import time
i = 0    # a counter
k = 5    # some shape parameter
# Delay start of stream by 5 sec (time to switch tabs)
time.sleep(5)
import requests
while True:
    x = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')
    y = (np.cos(k*i/50.)*np.cos(i/50.)+np.random.randn(1))[0]
    # Send data to your plot
    s.write(dict(x=x, y=y))
    time.sleep(1) # plot a point every second
# Close the stream when done plotting
s.close()
```

建立即時分析儀表板

如何建立一個即時監控儀表板



模擬感測器資料

```
with open('/tmp/machine.log', 'wa') as f:
    while True:
        recnum = random.randint(3,100)
        time.sleep(1)
        for i in range(1, recnum+1):
            machine_id = random.choice('12345')
            log_message = random.choice(['normal', 'warning', 'info', 'error'])
            log_value = random.randint(1,99)
            f.write('%s,%s,%s\n'%(machine_id, log_message, log_value))
        f.flush()
```

打入 python logGenerator.py
啟動 Log 產生程序

搜集Log資料

■ 觀看Log 資料

- ▣ `tail -f /tmp/machine.log`

■ 搜集最新Log

- ▣ `tail -f /tmp/machine.log | nc -lk localhost 9999`

資料會被導入9999 Port上的服務

建立 mytable

```
import happybase
connection = happybase.Connection('localhost', autoconnect=False)

connection.open()
connection.create_table(
    'mytable',
    { 'cf1': dict(),
      }
)

connection.close()
```


執行Spark 工作

■ 啟用Thrift 在 9090 上

- `/usr/hdp/current/hbase-master/bin/hbase-daemon.sh
start thrift -p 9090`

■ 啟用Spark 工作

- `spark-submit errorMiner.py localhost 9999 1>1.log
2>1.err`

啟用Spark, Streaming 以及SQL Context

```
# Start Spark Context and SQL Context  
sc = SparkContext(appName="qoo")  
sc.setLogLevel("ERROR")  
sqlContext = SQLContext(sc)  
ssc          = StreamingContext(sc, 1)
```

讀取以及處理資料

Read Data from Text Stream

```
lines = ssc.socketTextStream(sys.argv[1], int(sys.argv[2]))
```

Process Data

定義Schema

```
fields = ("machine_id", "log_message", "log_value")
```

```
mlog = namedtuple( 'mlog', fields )
```

```
counts = lines.map(lambda line:line.split(','))\  
                .filter(lambda ele: 'error' in ele[1])\  
                .map( lambda rec: mlog( rec[0], rec[1], int(rec[2])))
```

聚合資料

```
def aggregateRecord(rdd):  
    print rdd.count()  
    if rdd.count() > 0:  
        dic = {}  
        # Data Aggregate  
        df = rdd.toDF()  
        df.registerTempTable("mlog")  
        agg = sqlContext.sql("SELECT machine_id, avg(log_value) FROM  
mlog group by machine_id")  
        for k, v in agg.collect():  
            dic[k] = v
```

使用Spark SQL

寫入資料進HBase

```
connection.open()
table = connection.table('mytable')
current_date = datetime.now()
dt = current_date.timetuple()
ts = time.mktime(dt)
```

資料Aggregation

```
# put data into hbase
table.put(str(int(ts)), {'cf1:m1' : str(dic.get('1', 0))})
table.put(str(int(ts)), {'cf1:m2' : str(dic.get('2', 0))})
table.put(str(int(ts)), {'cf1:m3' : str(dic.get('3', 0))})
table.put(str(int(ts)), {'cf1:m4' : str(dic.get('4', 0))})
table.put(str(int(ts)), {'cf1:m5' : str(dic.get('5', 0))})
connection.close()
```

檢查Hbase 中的資料

■ 進入HBase Shell

- hbase shell

■ 查詢資料

- scan 'mytable'

使用Flask建立API

```
def api():  
    connection = happybase.Connection('localhost', autoconnect=False)  
    connection.open()  
    table = connection.table('mytable')  
  
    current_date = datetime.now()  
    dt = current_date.timetuple()  
    ts = int(time.mktime(dt)) - 30000  
  
    res = []  
    for key, data in table.scan(row_start=str(ts)):  
        res.append({'time':key, 'data':data})  
  
    connection.close()  
  
    return jsonify(res[-1])
```

<http://192.168.233.133:5000/>

使用API 建立Dashboard

```
import requests
while True:
    res = requests.get('http://192.168.233.133:5000/')
    d = res.json()
    x = d['dt']
    y = d['data']
    # Send data to your plot
    s.write(dict(x=x, y=y))

    time.sleep(1) # plot a point every second
```


The background features a light blue hexagonal grid pattern. Overlaid on this is a large, faint, circular graphic composed of concentric rings and radial lines, resembling a stylized sun or a target. The text "THANK YOU" is centered in a bold, dark blue, sans-serif font.

THANK YOU