



MSDS 498 CAPSTONE

Dan Kuratko



MY PROJECT

PROJECT 3 CLOUD MAPREDUCE

◆ Project Objectives

- ◊ Implement Chapter 16 in *Python for Programmers* on MS Azure
- ◊ Port the same example to run on Google Cloud Platform

◆ Specific Objectives

- ◊ Manipulate SQL relational database
- ◊ Store tweets in a MongoDB NoSQL JSON database, visualize via Folium map
- ◊ Execute Hadoop MapReduce using MS Azure HDInsight and GCP
- ◊ Use Apache Spark to process data in mini-batches with Azure and GCP
- ◊ Visualize IoT data via publish/subscribe model

SQL and NoSQL





SQL

- ◆ Review of SQL fundamentals
- ◆ Book database example
- ◆ SQLite library within Python

titles Table

```
6]: pd.read_sql('SELECT * FROM titles', connection)
```

	isbn	title	edition	copyright
0	0135404673	Intro to Python for CS and DS	1	2020
1	0132151006	Internet & WWW How to Program	5	2012
2	0134743350	Java How to Program	11	2018
3	0133976890	C How to Program	8	2016
4	0133406954	Visual Basic 2012 How to Program	6	2014
5	0134601548	Visual C# How to Program	6	2017
6	0136151574	Visual C++ How to Program	2	2008
7	0134448235	C++ How to Program	10	2017
8	0134444302	Android How to Program	3	2017
9	0134289366	Android 6 for Programmers	3	2016

NoSQL

- ◆ Create a MongoDB Atlas Cluster
- ◆ Tweet Streaming with Tweepy
- ◆ Use Folium maps to visualize tweet density



Dan's Capstone Project

Consumer Keys

On 01/12/2021 your consumer keys will no longer be visible.

API Key & Secret

Authentication Tokens

Bearer Token

Access Token & Secret

Helpful docs

How to use projects

App permissions

Authentication overview

Authentication best practices

Using bearer tokens

Using access token & secret

Keys & tokens let us know who you are.

Specifically, keys are unique identifiers that authenticate your App's request, while tokens are a type of authentication for an App to gain specific access to data.

SANDBOX

Dan's First Cluster

Version 4.2.9

CONNECT METRICS COLLECTIONS

CLUSTER TIER M0 Sandbox (General)

REGION North California (usenorth)

TYPE Replica Set - 3 nodes

LINKED REALM APP None Linked

Operations R: 0 W: 0

Latency

Connections 2

Last 4 Hours

Hadoop MapReduce

With MS Azure HDInsight



MapReduce Word Count

- ◆ Create Azure HDInsight multi-node cluster
 - ◆ Run Hadoop MapReduce on this cluster
 - ◆ Case study: Romeo and Juliet word count

KuratkoCluster | Cluster size

Overview Save Discard

Automatically increase or decrease the number of worker nodes based on a schedule or specific performance metrics. [Learn More](#)

This estimate does not include subscription discounts or costs related to storage, networking, or data transfer.

Actual costs vary based on realtime core usage.

Node type	Node size	Number of nodes	Estimated cost/hour
Head node	D9 v2 (4 Cores, 14 GB RAM), 0.29 USD/Hour	2	0.59 USD
Worker node	D9 v2 (4 Cores, 14 GB RAM), 0.29 USD/Hour	2	0.59 USD

Enable autoscale

TOTAL COST: 1.35 USD/HOUR (ESTIMATE)

Cluster size history

Oct 20 Oct 21 Oct 22 Oct 23 Oct 24 Oct 25 Oct 26 Oct 27 Oct 28 Oct 29 Oct 30 Oct 31 Nov 1 Nov 2 Nov 3 Nov 4

Number of Active Worker Nodes

2

```
[1] ➜ /Users/krishna/Downloads -- cd ./adobe/
```

```
File Input Format Counters
    Bytes Read:504800
File Output Format Counters
    Bytes Written:102
```

```
28/10/04 22:16:09 INFO streaming.StreamJob: Output
[1] ➜ /Users/krishna/Downloads -- rm -rf ./adobe/
```

```
28/10/04 22:16:12 INFO file.Lexer: LexicalAnalyzer: Line
28/10/04 22:16:12 INFO file.Lexer: LineIndex: Successfully
[1] ➜ rm -rf ./adobe/
```

```
1      1000
2      0
3      0
4      6828
5      3941
6      2997
7      1991
8      1337
9      1848
10     490
11     285
12     180
13     55
14     21
15     13
16     7
17     6
18     3
19     1
20     1
21     1
22     1
23     1
24     2
25     1
26     1
27     1
28     1
29     1
```

```
[1] ➜ rm ./adobe/
```

Apache Spark

Employing a Docker stack





MapReduce with Spark

- ◆ Execute a Docker stack with Spark and PySpark
- ◆ Run Hadoop MapReduce on this cluster
- ◆ Case study: Romeo and Juliet word count



The screenshot shows the Microsoft Azure portal interface. In the top navigation bar, 'dkcluster' is selected under 'Home > DK_second_cluster > dkcluster'. The main area displays the 'Activity log' for the HDInsight cluster. It includes sections for 'Overview', 'Activity log' (which is currently selected), 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Quick start', 'Tools', and 'Settings'. Below these sections, there is a table with two items:

Operation name	Status	Time	Time
> Create or Update Cluster	Succeeded	4 hours ago	Sun
> RunUserScriptActions	Succeeded	5 hours ago	Sun

```
(1) import findspark
findspark.init('/Users/dk/spark-2.4.7-bin-hadoop2.7')
import pyspark

(2) import nltk
nltk.download('stopwords')
[nltk_data] Downloading package stopwords to /Users/dk/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
(2) True

(3) from nltk.corpus import stopwords
stop_words = stopwords.words('english')

(4) from pyspark import SparkContext
configuration = SparkConf().setAppName('RomeoAndJulietCounter')
sc = SparkContext(conf=configuration)

(5) from operator import add
word_counts = sc.textFile('romeo_and_juliet.txt')\
    .map(lambda line: strip_punc(line).lower())\
    .map(lambda line: line.split())
[...]
```

```
(7) filtered = tokenized.filter(lambda word: word not in stop_words)

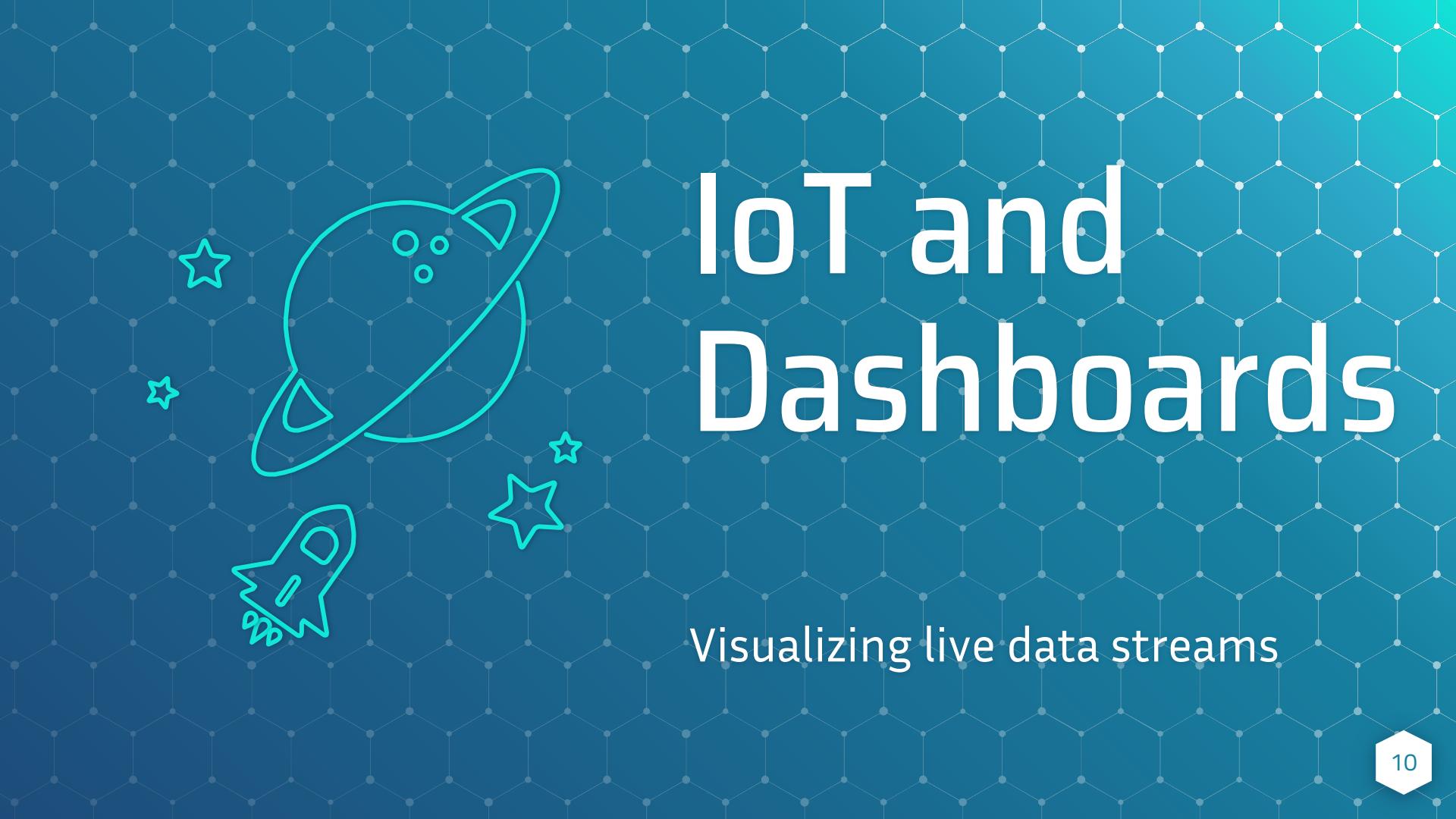
(8) from operator import add
word_counts = filtered.map(lambda word: (word, 1)).reduceByKey(add)

(9) sorted_items = word_counts.collect()
sorted_items = sorted_items.sortByKey(keyType='reversed')

(10) for word, count in sorted_items:
    print(f'{word}:{count}')
```

```
[13]: max_len = max([len(word) for word, count in sorted_items])
       for word, count in sorted_items:
           print(f'{word}:{count}')
```

word	count
romeo	303
thou	277
juliet	183
thy	170
nurse	146
capulet	141
love	136
thee	135
shall	112
lady	109
friar	104
come	95
project	90
mercurio	83
good	80
benvolio	79
enter	75
go	75
i'll	71
tybalt	69
death	69
night	68
lawrence	67
man	65
hath	64
may	63
one	61



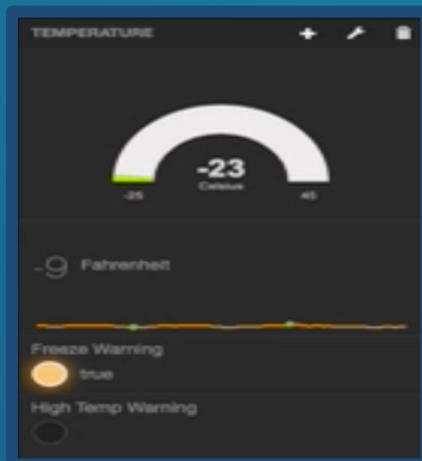
IoT and Dashboards

Visualizing live data streams



Live Data Visualization

- ◆ Visualize simulated PubNub data with Freeboard.io
- ◆ Simulate IoT thermostat using JSON "dweets"





Hadoop with Google

Replicating a MapReduce in GCP

MapReduce with GCP

- ◆ GCP Dataproc cluster, composer environment, storage bucket
- ◆ Word count harnessing GCP and Apache Airflow

Running a Hadoop wordcount job on a Dataproc cluster

Running a Hadoop wordcount job on a Dataproc cluster

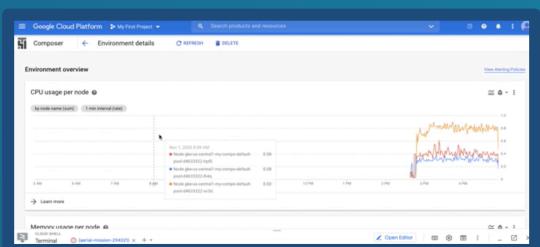
About this codelab

Last updated Oct 29, 2020

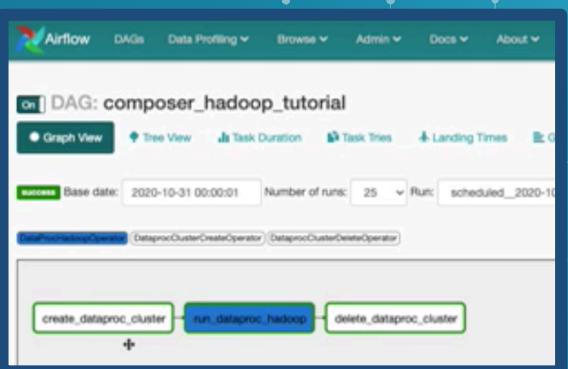
Written by a Googler

1. Introduction

Workflows are a common use case in data analysis - they involve ingesting, transforming, and analyzing data to find the meaningful information within. In Google Cloud Platform, the tool for orchestrating workflows is Cloud Composer, which is a hosted version of the popular open source workflow tool Apache Airflow. In this lab, you will use Cloud Composer to create a simple workflow that creates a Cloud Dataproc cluster, analyzes it using Cloud Dataproc and Apache Hadoop, then deletes the Cloud Dataproc cluster afterwards.



OBJECTS	CONFIGURATION	PERMISSIONS	RETENTION	LIFECYCLE
Buckets > kuratkobucket				
UPLOAD FILES	UPLOAD FOLDER	CREATE FOLDER	MANAGE HOLDS	DELETE
<input type="checkbox"/> Filter: Filter by object or folder name prefix				
Wordcount				
length_mapper	text/plain	New 1, 2020, 1...	Standard	
length_reducer	text/plain	New 1, 2020, 1...	Standard	
runner.py	text/python-script	New 1, 2020, 1...	Standard	
wordcount/	Folder			





Spark and GCP

Apache Spark word count in GCP

Spark MapReduce with GCP

- ◆ Same word count example using Apache Spark

Dataproc > Documentation > Guides

Rate and review  

Use the Cloud Storage connector with Apache Spark

Table of contents ▾

- Objectives
- Costs
- Before you begin
 - Prepare the Spark wordcount job
 - Submit the job
- ***

This tutorial shows you how to run example code that uses the Cloud Storage connector with Apache Spark .

The screenshot shows the Google Cloud Platform interface with the 'Dataflow' project selected. The main title is 'Job details' for job ID '49520619855341e7a97149d0fb14456'. The job was started on Nov 8, 2020, at 7:35:12 PM, with a duration of 34 sec and a status of 'Success'. Below the title, there are tabs for 'Output' and 'Configuration', with 'Configuration' currently selected. Under the 'Edit' section, the configuration includes:

- Region: us-central1
- Cluster: sparkexample
- Job type: PySpark
- Main python file: gs://spark-staging-us-central1/49520619855341e7a97149d0fb14456/staging/wordcount.py

Additional python files listed are 'Jar files', 'Properties', and 'Arguments'. The 'Arguments' section contains two entries: 'gs://ik_spark_2_bucket/input/' and 'gs://ik_spark_2_bucket/output/'. The bottom of the page features a 'REFRESH' button and a 'CLONE' link.

```
yarnApplications:
- name: word-count.py
  progress: 1.0
  state: FINISHED
  trackingUrl: http://sparkexample-m:8888/pro
(base) Dens-MacBook-Air:Downloads dks gutil
('u'a', 23)
('u'm', 21)
('u'would', 1)
('u'What's', 1)
('u'sweet', 1)
('u'as', 1)
('u'call', 1)
('u'which', 1)
('u'name', 1)
('u'That', 1)
('u'reee', 1)
('u'any', 1)
('u'other', 1)
('u'in', 1)
('u'name?', 1)
('u'By', 1)
(base) Dens-MacBook-Air:Downloads dks
```

THANK YOU!

Dan Kuratko

dankuratko@gmail.com

(619)-618-5093

