L07: SQL & SparkSQL

ANLY 502: Massive Data Fundamentals

Simson Garfinkel & Marck Vaisman

March 27, 2017

Outline for today's class — Databases!

Administrivia

Final projects

Databases

- Database data models: records, tables, relational, object, document
- Database access models: Embedded, Client/Server, Distributed
- Database persistence/durability models: ephemeral, durable, streaming
- Database options at Amazon

SQL

- Flavors of SQL
- Database Normalization
- Worked example: tracking files with SQLite.

Lab:

- Accessing Forensicswiki logs in SQL server

SparkSQL

- Building a database from text files
- Building a database from MySQL

Hive

Administrivia

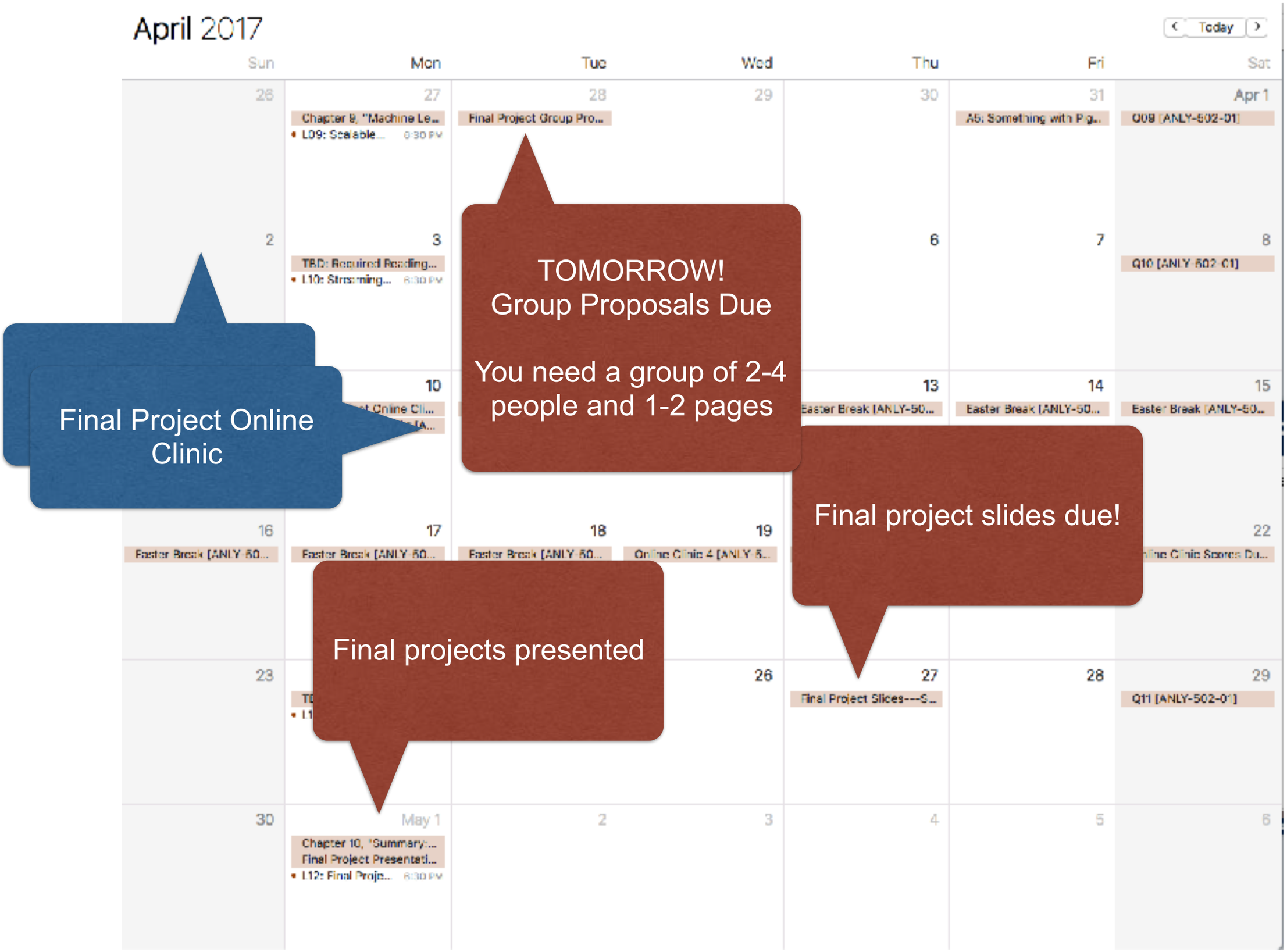
A5 — It's out!

A4 — It's graded!

Project Groups — Due tomorrow!

- Email your group NAME and MEMBERSHIP to anly502@nitroba.org
- Your group will be created on Canvas
- Submit your proposal by Wednesday 5pm
- We will comment on your proposals!

1 Week March + 4 Weeks in April 2017 + 1 day in May!



Connecting to Jupyter & Hue web-based interfaces
SparkSQL
Hive

Part 2

Sizing your cluster

How big should your cluster be?

Most of our clusters have been 1 node:

- Spark Master
- HDFS Manager
- Job distribution
- Many JVMs.
- ...

The m3.xlarge is not large enough for this!

Amazon EMR VMs are configured without virtual memory.

- Processes fail to start
- Java processes freeze, waiting for more memory
- Some processes are killed.

So we contacted Amazon Tech Support and asked:

- Why do you support m4.xlarge if it isn't large enough?

Amazon to ANLY502: you need to use bigger clusters

Remember — Amazon is running Spark on top of Yarn

yarn.nodemanager.resource.memory-mb memory allocated per physical container	6144M	/etc/hadoop/conf/yarn-site.xml
spark.executor.memory	5120M	/etc/spark/conf/spark-defaults.conf
spark.yarn.am.memory	512M	pyspark runs by default in "client" deploy mode
spark.yarn.am.memoryOverhead	384M	
spark.yarn.executor.memoryOverhead	512M	

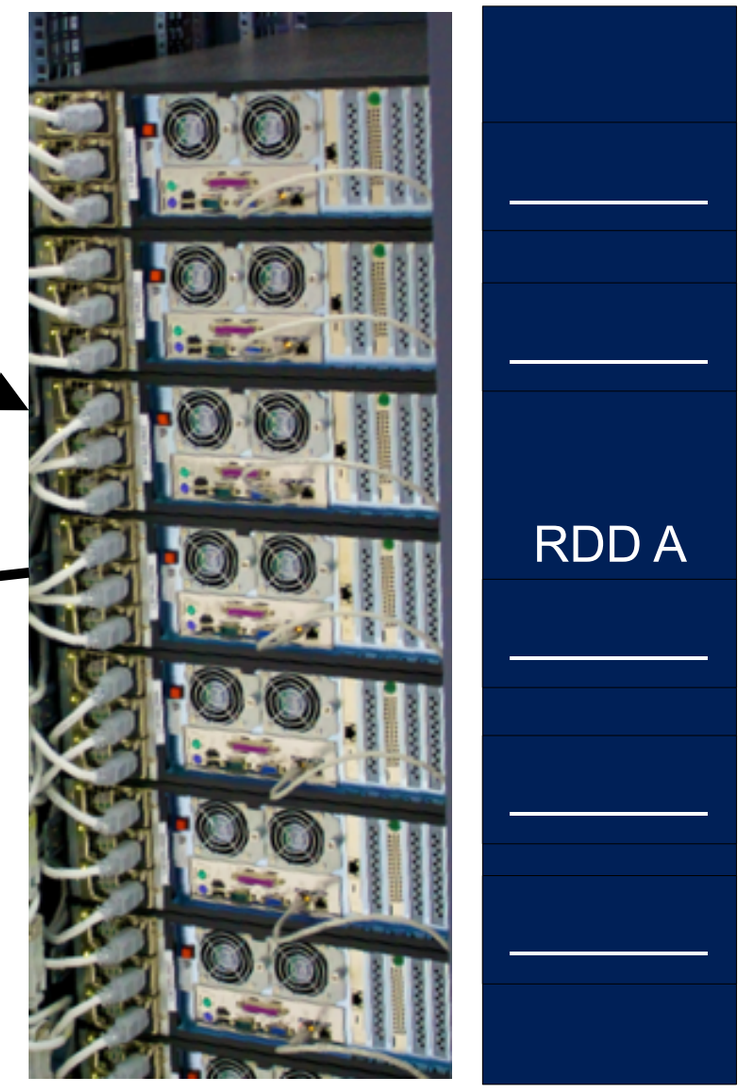
After executor was allocated, there wasn't sufficient memory.

See http://spark.apache.org/docs/latest/running-on-yarn.html

More about RDDs

A RDD may be on multiple nodes.

The control program is called the "Driver."



HadoopRDD
A = sc.textFile(fn)

```
A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")

print( B.count() )

def hasHamlet( s ):
    return "Hamlet" in s

B = A.filter( hasHamlet )
```

There are two ways for the "Driver" to run on YARN

cluster mode — The Driver runs inside an application master (AM)

- managed by YARN.

client mode — The driver runs inside the client process.

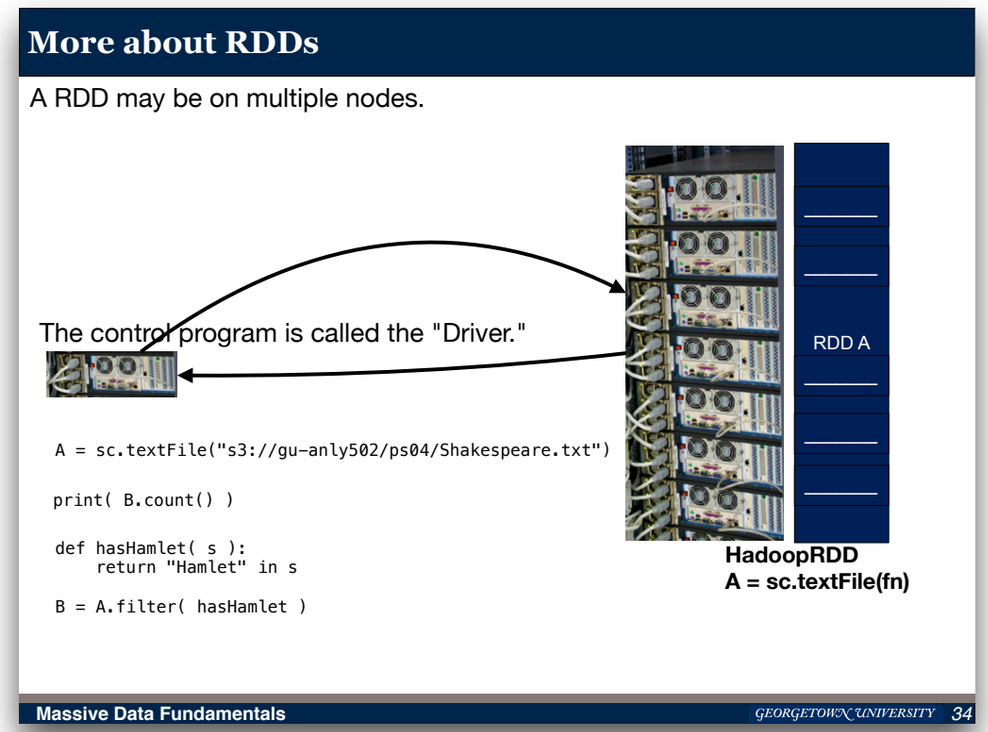
- The AM requests YARN resources for the workers.

AWS runs pyspark in "client" mode.

- Driver always runs
- Additional YARN resources not always available!

AWS runs with log aggregation enabled.

- Logs are collected at the head from other nodes.
- Debug with "yarn logs -applicationID <app ID>"



Information on debugging:

- http://spark.apache.org/docs/latest/running-on-yarn.html#debugging-your-application

Information on configuration on AWS:

- http://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-configure-apps.html

From Amazon:

"EMR offers a huge amount of configuration option compared to our other managed services."

- Intended for advance users who require customizations.
- Only Amazon service where you can **ssh** into the EC2 instances and have full root access.
- Presume that you know what you are doing.

m3.xlarge instance is the default because it offers balance resources.

- Not appropriate in every case, why is why other options are available.
- **r3.*, r4.*, m2.*, cr1.*** types provide more memory

See how much memory is available with this table:

- http://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hadoop-task-config.html

Compare m3.xlarge and r4.xlarge:
(Note: R4 is only available on EMR 5.4.0 and later)

m3.xlarge

Configuration Option	Default Value	With HBase Installed
mapreduce.map.java.opts	-Xmx1152m	-Xmx1152m
mapreduce.reduce.java.opts	-Xmx2304m	-Xmx2304m
mapreduce.map.memory.mb	1440	1440
mapreduce.reduce.memory.mb	2880	2880
yarn.app.mapreduce.am.resource.mb	2880	2880
yarn.scheduler.minimum-allocation-mb	32	32
yarn.scheduler.maximum-allocation-mb	11520	5760
yarn.nodemanager.resource.memory-mb	11520	5760

r4.xlarge

Configuration Option	Default Value	With HBase Installed
mapreduce.map.java.opts	-Xmx4685m	-Xmx2342m
mapreduce.reduce.java.opts	-Xmx9370m	-Xmx4684m
mapreduce.map.memory.mb	5856	5856
mapreduce.reduce.memory.mb	11712	11712
yarn.app.mapreduce.am.resource.mb	11712	11712
yarn.scheduler.minimum-allocation-mb	32	32
yarn.scheduler.maximum-allocation-mb	23424	11712
yarn.nodemanager.resource.memory-mb	23424	11712

With HBase installed, there is less memory for Spark.

-Xmxsize controls the size of the Java memory pool!

```
$ man java

-Xmxsize

Specifies the maximum size (in bytes) of the memory allocation pool
in bytes. This value must be a multiple of 1024 and greater than 2
MB. Append the letter k or K to indicate kilobytes, m or M to
indicate megabytes, g or G to indicate gigabytes. The default value
is chosen at runtime based on system configuration. For server
deployments, -Xms and -Xmx are often set to the same value. See the
section "Ergonomics" in Java SE HotSpot Virtual Machine Garbage
Collection Tuning Guide at http://docs.oracle.com/javase/8/docs/
technotes/guides/vm/gctuning/index.html.

The following examples show how to set the maximum allowed size of
allocated memory to 80 MB using various units:

      -Xmx83886080
      -Xmx81920k
      -Xmx80m

The -Xmx option is equivalent to -XX:MaxHeapSize.
```

Amazon on HBase: don't use it!

With HBase installed, there is less memory available to YARN applications.

If you need NoSQL database, look at DynamoDB — it offers numerous advantages over HBase.

Recommend that you read "Plan and Configure Clusters"

- What region to run the cluster in, where and how to store data, and how to output results:
  - *http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan.html*
- Long-Running vs. Transient Clusters
  - *http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-longrunning-transient.html*
- What software you run:
  - *http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-software.html*
- Choosing the hardware and networking options:
  - *http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-instances.html*
- More:
  - *http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan.html*

Running single-node clusters is not recommended (by Amazon)

Runs all master daemons on the master:

- namenode, clustermanager, base master, hue, etc.

Runs all of the slave daemons on the master:

- tasknode, nodemanager, regionserver

One a 2+ node cluster:

- the master only runs the master daemons
- The Core nodes run all the salve daemons
- The Task nodes don't run node manager because they are not part of HDFS.

Bottom line: If you are going to run HBase:

- You need core nodes
- And probably task nodes.

For more information

Amazon Whitepaper, EMR Best Practices:

- https://aws.amazon.com/whitepapers/emr-best-practices/

Specific scenarios for:

- Moving large amounts of data from HDFS to S3
- Moving large amounts of local data to S3
- Moving large amounts of data from S3 to HDFS
- Data Collection
- Data Aggregation
- Processing data with EMR
- Optimizing for Cost
- Performance Optimization

Amazon Web Services – **Best Practices for Amazon EMR** August 2013

amazon web services

Best Practices for Amazon EMR

Parviz Deyhim

August 2013

(Please consult <http://aws.amazon.com/whitepapers/> for the latest version of this paper)

Page 1 of 38

amazon web services

Spark SQL

More stuff!

Spark and SparkSQL— what you need to know at this point

Spark computes with RDD — Resilient Distributed Datasets

- RDD consists of records.
- Records can be interpreted as Key/Value pairs.

RDDs are arranged in a directed, acyclic graph (DAG)

RDDs are evaluated with "lazy evaluation" — only when needed.

RDDs can be created by:

- Reading text files from HDFS or S3 (**sc.textFile(), sc.wholeTextFiles(), etc.)**
- Loading a saved RDD (**RDD.saveAsPickleFile** and **sc.pickleFile()**)
- Operations on other RDDs

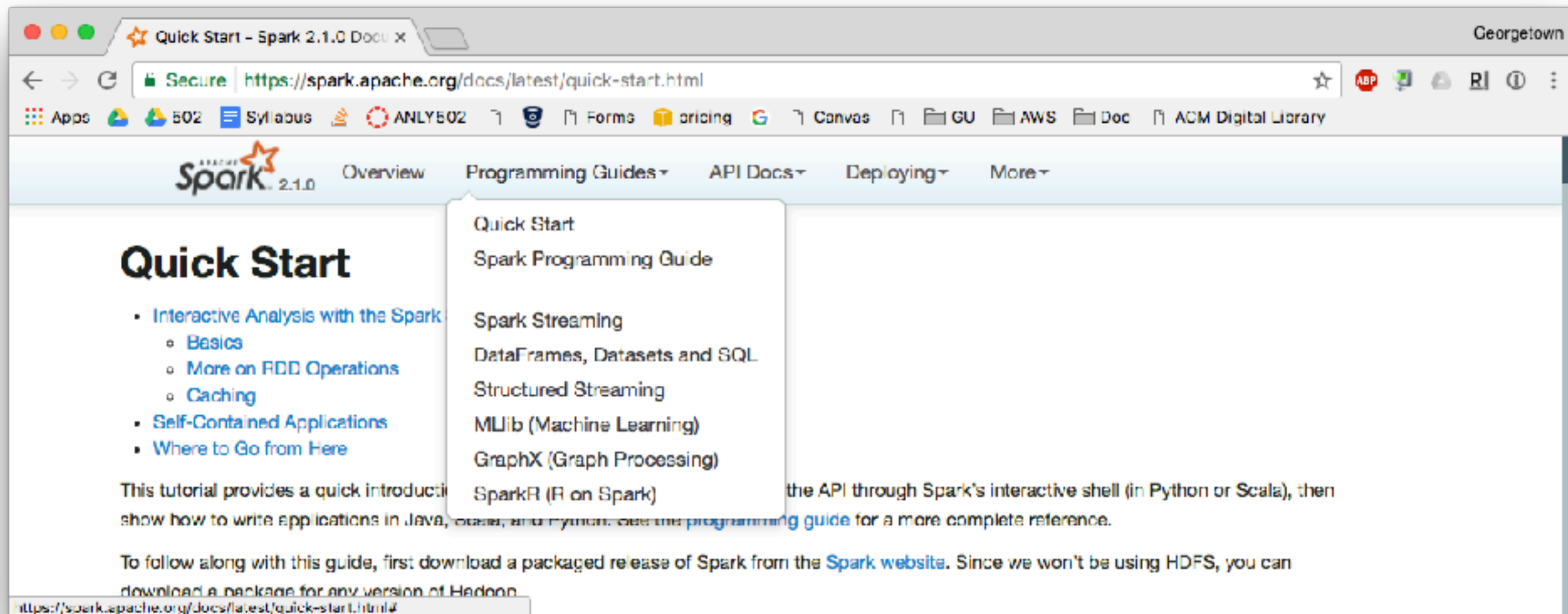
DataFrames store rows of structured data (Datasets in Scala and Java)

- Spark API: `df.select(df['name'], df['age']).filter("df['age'] > 21).show()`
- SQL API: `spark.sql("select name, age from ages where age>21").show()`

Feeling lost? Be sure to *read the documentation.*

Programming guides:

- Quick Start
- Spark Programming Guide
- DataFrames, Datasets and SQL



Running Python & Spark

ipyspark — Revised in bootstrap2-ipython.sh:

- Environment variables:

```
PYSPARK_PYTHON=python3
PYSPARK_DRIVER_PYTHON=ipython3
```

- Executable:

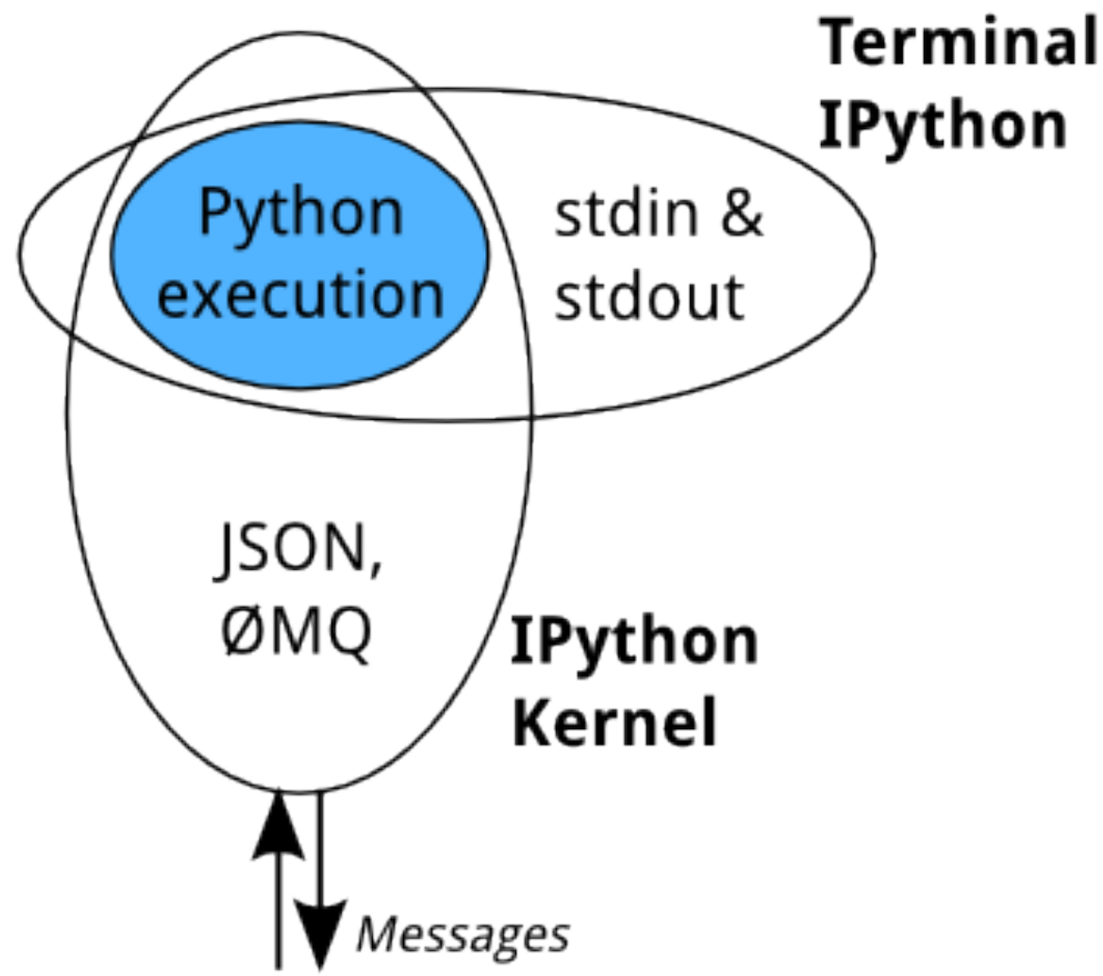
```
pyspark
```

IPython combines a REPL and a kernel

- Sort of like this:

```
while True:
    code = input(">>> ")
    print(eval(code))
```

- The Python kernel runs in a different process.
- The same kernel is used for Jupyter and Qt.



What's going on with pyspark?

pyspark

- runs on the Master — Only installed on the head-end
- Communicates with the Spark Driver
- Sets up environment variables for iPython and then runs spark-submit.
- spark-submit runs Java (JRE) and the Scala Spark runtime, connected to IPython kernel
- IPython kernel communicates with Scala using ZMQ (0MQ)

pyspark uses Py4J to bridge between Python and Java

- Allows Python programs to dynamically access Java objects in the Java Virtual Machine.
- Lets Java collections be used as Python collections
- Allows call-backs, so Java programs can call Python objects.

What's going on with pyspark?

Best way to learn about a program — read the documentation:

```
[hadoop@ip-172-31-41-157 ~]$ pyspark --help
Usage: ./bin/pyspark [options]

Options:
  --master MASTER_URL         spark://host:port, mesos://host:port, yarn, or local.
  --deploy-mode DEPLOY_MODE   Whether to launch the driver program locally ("client") or
                              on one of the worker machines inside the cluster ("cluster")
                              (Default: client).
  --class CLASS_NAME          Your application's main class (for Java / Scala apps).
  --name NAME                 A name of your application.
  --jars JARS                 Comma-separated list of local jars to include on the driver
                              and executor classpaths.
  --packages                  Comma-separated list of maven coordinates of jars to include
                              on the driver and executor classpaths. Will search the local
                              maven repo, then maven central and any additional remote
                              repositories given by --repositories. The format for the
                              coordinates should be groupId:artifactId:version.
  --exclude-packages          Comma-separated list of groupId:artifactId, to exclude while
                              resolving the dependencies provided in --packages to avoid
                              dependency conflicts.
```

pyspark is running a JVM *and* a Python interpreter

```
  --py-files PY_FILES         Comma-separated list of .zip, .egg, or .py files to place
                              on the PYTHONPATH for Python apps.
  --files FILES               Comma-separated list of files to be placed in the working
                              directory of each executor.

  --conf PROP=VALUE           Arbitrary Spark configuration property.

  --driver-memory MEM         Memory for driver (e.g. 1000M, 2G) (Default: 1024M).
  --driver-java-options       Extra Java options to pass to the driver.
  --driver-library-path       Extra library path entries to pass to the driver.
  --driver-class-path         Extra class path entries to pass to the driver. Note that
                                jars added with --jars are automatically included in the
                                classpath.
  YARN-only:
  --driver-cores NUM          Number of cores used by the driver, only in cluster mode
                              (Default: 1).
  --queue QUEUE_NAME          The YARN queue to submit to (Default: "default").
  --num-executors NUM         Number of executors to launch (Default: 2).
                              If dynamic allocation is enabled, the initial number of
                              executors will be at least NUM.
  --archives ARCHIVES         Comma separated list of archives to be extracted into the
                              working directory of each executor.
  --principal PRINCIPAL       Principal to be used to login to KDC, while running on
                              secure HDFS.
  --keytab KEYTAB             The full path to the file that contains the keytab for the
                              principal specified above. This keytab will be copied to
                              the node running the Application Master via the Secure
                              Distributed Cache, for renewing the login tickets and the
                              delegation tokens periodically.
```

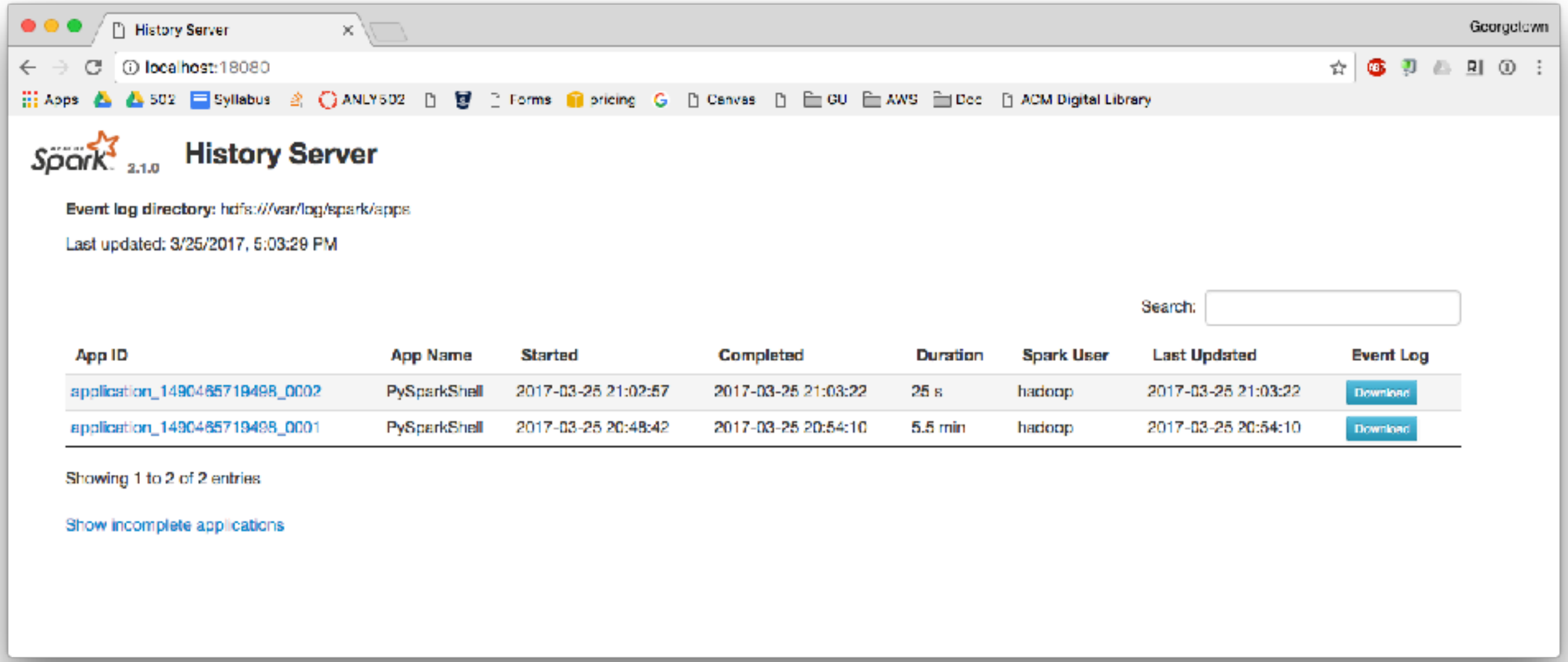
spark-submit — the "non-interactive" version.

pyspark — creates the SparkSession and context objects.

spark-submit — You need to create them yourself

```
#!/usr/bin/env python34
# A5/demo.py
# Run this program with: $ spark-submit demo.py

import sys,os,datetime,re,operator
from pyspark.sql import SparkSession

if __name__=="__main__":
    spark = SparkSession.builder.appName("quazyilx").getOrCreate()
    sc    = spark.sparkContext

    sc.setLogLevel("ERROR")                    # Lower logging

    print( "You are using Spark {}".format( spark.version ) )

    # Put the numbers 1 to a million in an RDD and add them
    rdd = sc.parallelize( range( 1, 1000001 ) )

    print( "The numbers 1 to a million added together are: {}".
            format( rdd.reduce( operator.add ) ) )

    # Print the current time with SQL
    now = spark.sql("select now()").take(1)[0][0]

    print("Today is: {}".format( now.isoformat() ) )

    spark.stop()
```

Spark History Server: port 18080



$ ssh -A -L18080:localhost:18080 hadoop@*remotehostname*

Look at stats on logfile…

```
$ ipyspark --py-files=fwiki.py
Python 3.4.3 (default, Sep  1 2016, 23:33:38)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/03/25 22:41:36 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set,
falling back to uploading libraries under SPARK_HOME.
Welcome to

      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.1.0
      /_/

Using Python version 3.4.3 (default, Sep  1 2016 23:33:38)
SparkSession available as 'spark'.

In [1]: logs = sc.textFile("s3://gu-anly502/logs/forensicswiki.2012.txt")

In [2]: logs.take(1)
Out[2]: ['77.21.0.59 - - [01/Jan/2012:00:35:03 -0800] "GET /wiki/Write_Blockers HTTP/1.1"
200 5742 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.52.7 (KHTML,
like Gecko) Version/5.1.2 Safari/534.52.7" ']

In [3]:
```

History Server

localhost:18080

Apps 502 Syllabus ANLY502 Forms pricing Canvas GU AWS Doc ACM Digital Library

Spark 2.1.0

History Server

Event log directory: hdfs:///var/log/spark/apps

Last updated: 3/25/2017, 6:43:16 PM

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1490465719498_0004	PySparkShell	2017-03-25 22:37:54	2017-03-25 22:40:19	2.4 min	hadoop	2017-03-25 22:40:19	Download
application_1490465719498_0003	PySparkShell	2017-03-25 21:56:48	2017-03-25 22:37:49	41 min	hadoop	2017-03-25 22:37:49	Download
application_1490465719498_0002	PySparkShell	2017-03-25 21:02:57	2017-03-25 21:03:22	25 s	hadoop	2017-03-25 21:03:22	Download
application_1490465719498_0001	PySparkShell	2017-03-25 20:48:42	2017-03-25 20:54:10	5.5 min	hadoop	2017-03-25 20:54:10	Download

Showing 1 to 4 of 4 entries

Show incomplete applications

PySparkShell - Spark Jobs

localhost:18080/history/application_1490465719498_0004/jobs/

Apps 502 Syllabus ANLY502 Forms pricing Canvas GU AWS Doc ACM Digital Library

Spark 2.1.0 | Jobs | Stages | Storage | Environment | Executors | PySparkShell application UI

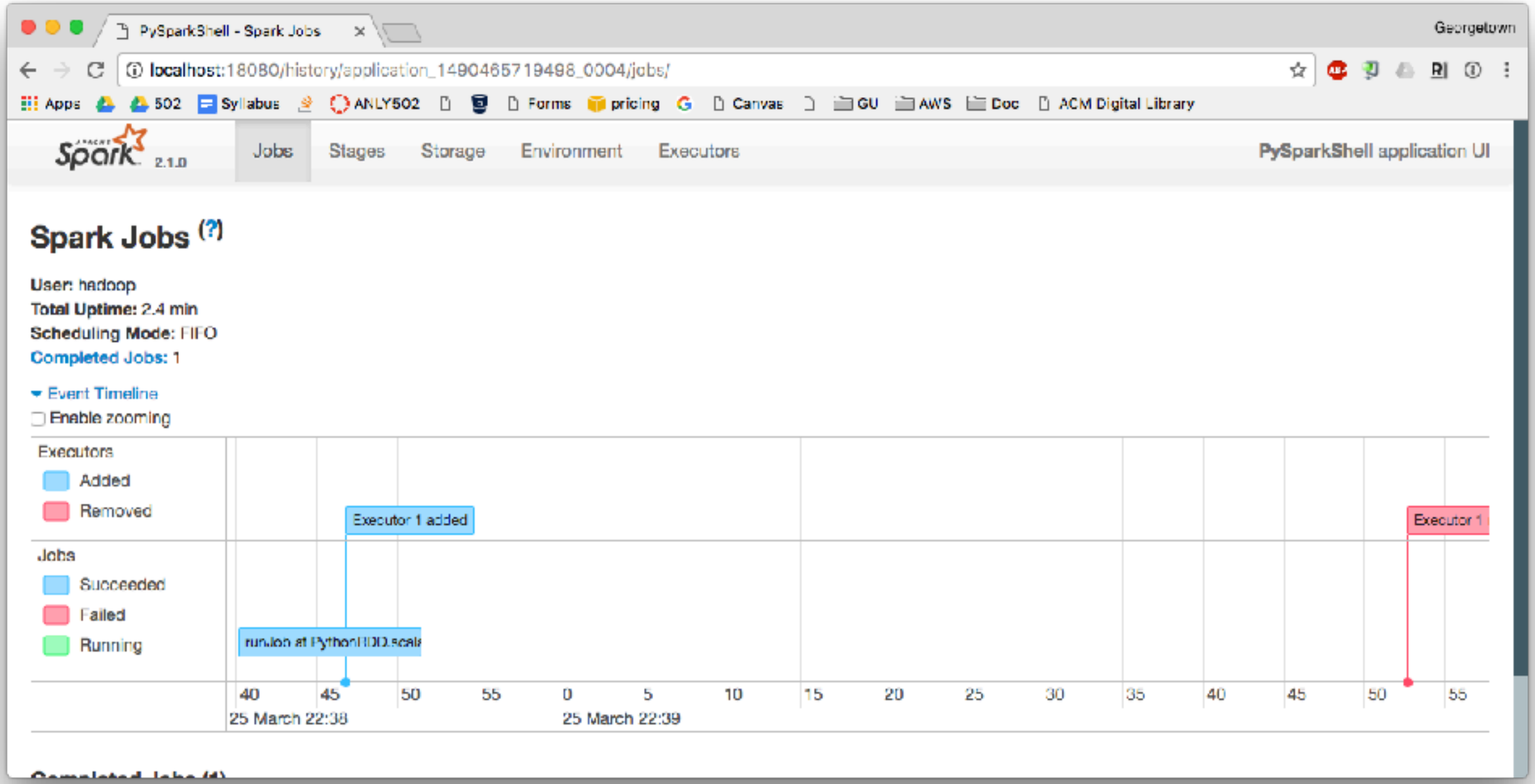
Spark Jobs (?)

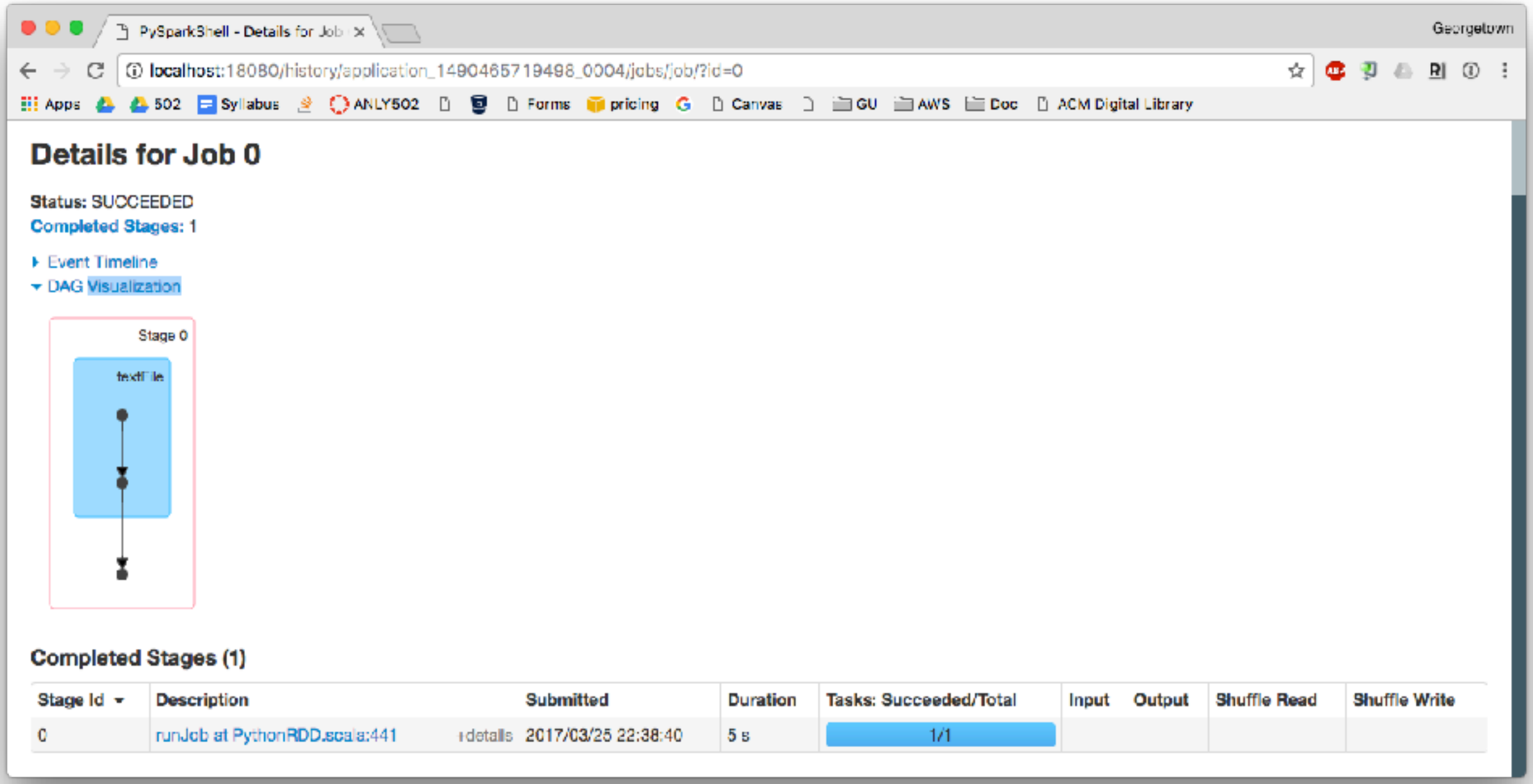
User: hadoop
Total Uptime: 2.4 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	runJob at PythonRDD.scala:441	2017/03/25 22:38:40	11 s	1/1	1/1


PySparkShell - Spark Jobs
Georgetown
localhost:18080/history/application_1490465719498_0004/jobs/
Apps
502
Syllabus
ANLY502
Forms
pricing
Canvas
GU
AWS
Doc
ACM Digital Library
Spark 2.1.0
Jobs
Stages
Storage
Environment
Executors
PySparkShell application UI
Spark Jobs (?)
User: hadoop
Total Uptime: 2.4 min
Scheduling Mode: FIFO
Completed Jobs: 1
Event Timeline
Enable zooming
Executors
Added
Removed
Executor 1 added
Executor 1
Jobs
Succeeded
Failed
Running
40
45
50
55
0
5
10
15
20
25
30
35
40
45
50
55
25 March 22:38
25 March 22:39


PySparkShell - Details for Job
Georgetown
localhost:18080/history/application_1490465719498_0004/jobs/job/?id=0
Apps
502
Syllabus
ANLY502
Forms
pricing
Canvas
GU
AWS
Doc
ACM Digital Library
Details for Job 0
Status: SUCCEEDED
Completed Stages: 1
Event Timeline
DAG Visualization
Stage 0
textFile
Completed Stages (1)
Stage Id
Description
Submitted
Duration
Tasks: Succeeded/Total
Input
Output
Shuffle Read
Shuffle Write
0
runJob at PythonRDD.scala:441
details
2017/03/25 22:38:40
5 s
1/1

```
In [1]: logs = sc.textFile("s3://gu-anly502/logs/forensicswiki.
2012.txt")

In [2]: logs.take(1)
Out[2]: ['77.21.0.59 - - [01/Jan/2012:00:35:03 -0800] "GET /wiki/
Write_Blockers HTTP/1.1" 200 5742 "-" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_6_8) AppleWebKit/534.52.7 (KHTML, like Gecko) Version/
5.1.2 Safari/534.52.7" ']

In [3]: import fwiki

In [4]: logObjects = logs.map(lambda line: fwiki.LogLine(line))

In [5]: logDF = spark.createDataFrame( logObjects.map( lambda obj:
obj.Row() ) )

In [6]: logDF.cache()
Out[6]: DataFrame[agent: string, bytes: bigint, datetime: timestamp,
ipaddr: string, method: string, path: string, refer: string, result:
bigint]

In [7]: logDF.count()
Out[7]: 15949554
```

```
In [8]: logDF.registerTempTable("log")

In [9]: spark.sql("select count(*) from log").show()
+--------+
|count(1)|
+--------+
|15949554|
+--------+

In [11]: spark.sql("select count(*) from log where month(datetime)=3").show()
+--------+
|count(1)|
+--------+
| 1279546|
+--------+

In [13]: spark.sql("select year(datetime),month(datetime),count(*) from log group by
year(datetime),month(datetime) order by 1,2").show()
+----------------------------+-----------------------------+--------+
|year(CAST(datetime AS DATE))|month(CAST(datetime AS DATE))|count(1)|
+----------------------------+-----------------------------+--------+
|                        null|                         null|     586|
|                        2012|                            1| 1531126|
|                        2012|                            2| 1327073|
|                        2012|                            3| 1279546|
|                        2012|                            4| 1014210|
|                        2012|                            5| 1169192|
|                        2012|                            6| 1304089|
|                        2012|                            7| 1282417|
|                        2012|                            8| 1453932|
|                        2012|                            9| 1281993|
|                        2012|                           10| 1499991|
|                        2012|                           11| 1395354|
|                        2012|                           12| 1397617|
|                        2013|                            1|   12428|
+----------------------------+-----------------------------+--------+

In [14]: quit();
```

Don't forget to quit()!

History Server

localhost:18080

Apps 502 Syllabus ANLY502 Forms pricing Canvas GU AWS Doc ACM Digital Library

Spark 2.1.0 **History Server**

Event log directory: hdfs:///var/log/spark/apps

Last updated: 3/25/2017, 7:06:37 PM

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1490465719498_0005	PySparkShell	2017-03-25 22:41:32	2017-03-25 23:06:30	25 min	hadoop	2017-03-25 23:06:30	Download
application_1490465719498_0004	PySparkShell	2017-03-25 22:37:54	2017-03-25 22:40:19	2.4 min	hadoop	2017-03-25 22:40:19	Download
application_1490465719498_0003	PySparkShell	2017-03-25 21:56:48	2017-03-25 22:37:49	41 min	hadoop	2017-03-25 22:37:49	Download
application_1490465719498_0002	PySparkShell	2017-03-25 21:02:57	2017-03-25 21:03:22	25 s	hadoop	2017-03-25 21:03:22	Download
application_1490465719498_0001	PySparkShell	2017-03-25 20:48:42	2017-03-25 20:54:10	5.5 min	hadoop	2017-03-25 20:54:10	Download

Showing 1 to 5 of 5 entries

Show incomplete applications

Only appears after you quit!

PySparkShell - Spark Jobs

localhost:18080/history/application_1490465719498_0005/jobs/

Spark 2.1.0 | Jobs | Stages | Storage | Environment | Executors | SQL | PySparkShell application UI

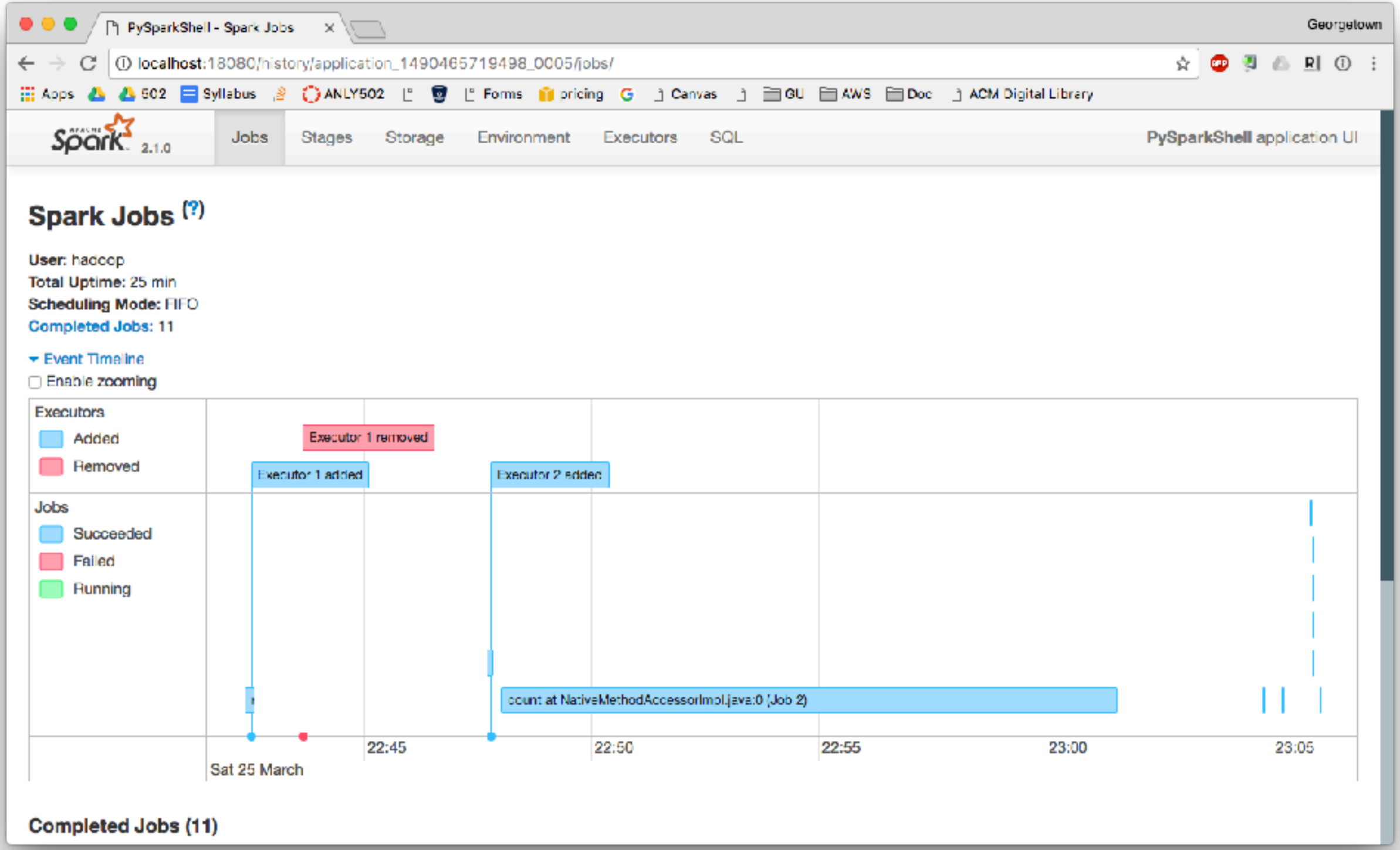
Spark Jobs (?)

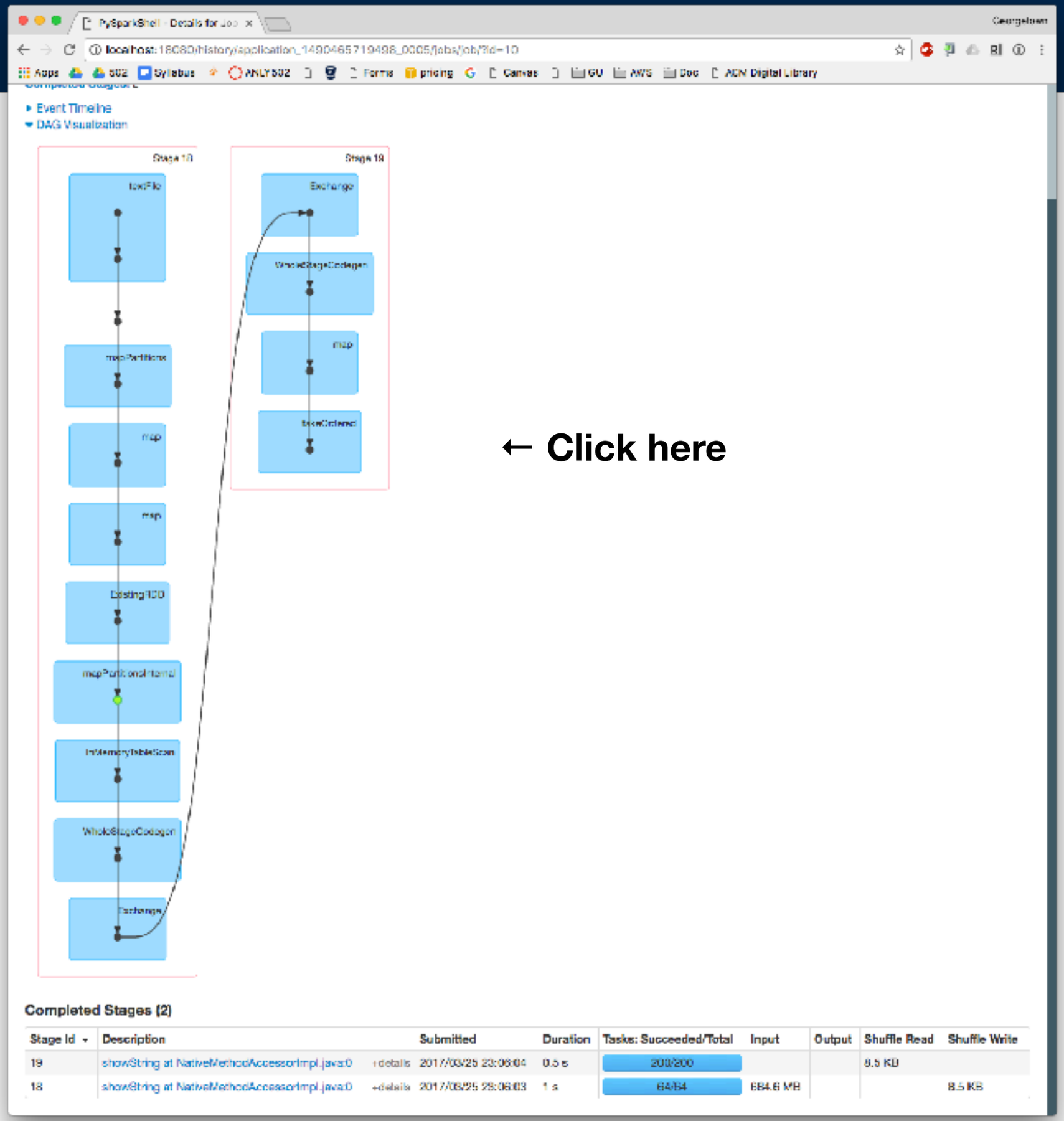
User: hadoop
Total Uptime: 25 min
Scheduling Mode: FIFO
Completed Jobs: 11

▸ Event Timeline

Completed Jobs (11)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
10	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:06:03	2 s	2/2	264/264
9	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:05:52	0.2 s	1/1 (1 skipped)	75/75 (64 skipped)
8	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:05:52	0.3 s	1/1 (1 skipped)	100/100 (64 skipped)
7	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:05:52	89 ms	1/1 (1 skipped)	20/20 (64 skipped)
6	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:05:52	31 ms	1/1 (1 skipped)	4/4 (64 skipped)
5	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:05:49	3 s	2/2	65/65
4	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:05:13	2 s	2/2	65/65
3	showString at NativeMethodAccessorImpl.java:0	2017/03/25 23:04:46	0.4 s	2/2	65/65
2	count at NativeMethodAccessorImpl.java:0	2017/03/25 22:48:01	14 min	2/2	65/65
1	runJob at PythonRDD.scala:441	2017/03/25 22:47:42	8 s	1/1	1/1
0	runJob at PythonRDD.scala:441	2017/03/25 22:42:21	13 s	1/1	1/1


PySparkShell - Spark Jobs
Georgetown
localhost:18080/history/application_1490465719498_0005/jobs/
Apps
502
Syllabus
ANLY502
Forms
pricing
Canvas
GU
AWS
Doc
ACM Digital Library
Spark 2.1.0
Jobs
Stages
Storage
Environment
Executors
SQL
PySparkShell application UI
Spark Jobs (?)
User: hadoop
Total Uptime: 25 min
Scheduling Mode: FIFO
Completed Jobs: 11
Event Timeline
Enable zooming
Executors
Added
Removed
Executor 1 removed
Executor 1 added
Executor 2 added
Jobs
Succeeded
Failed
Running
count at NativeMethodAccessorImpl.java:0 (Job 2)
22:45
22:50
22:55
23:00
23:05
Sat 25 March
Completed Jobs (11)


Event Timeline
DAG Visualization
Stage 18
Stage 19
textFile
Exchange
WholeStageCodegen
map
textFile
map
map
Exchange
Completed Stages (2)
Stage Id
Description
Submitted
Duration
Tasks: Succeeded/Total
Input
Output
Shuffle Read
Shuffle Write
19
showString at NativeMethodAccessorImpl.java:0
2017/03/25 23:06:04
0.5 s
200/200
8.5 KB
18
showString at NativeMethodAccessorImpl.java:0
2017/03/25 23:06:03
1 s
64/64
884.6 MB
8.5 KB

← **Click here**

PySparkShell - Details for Stage

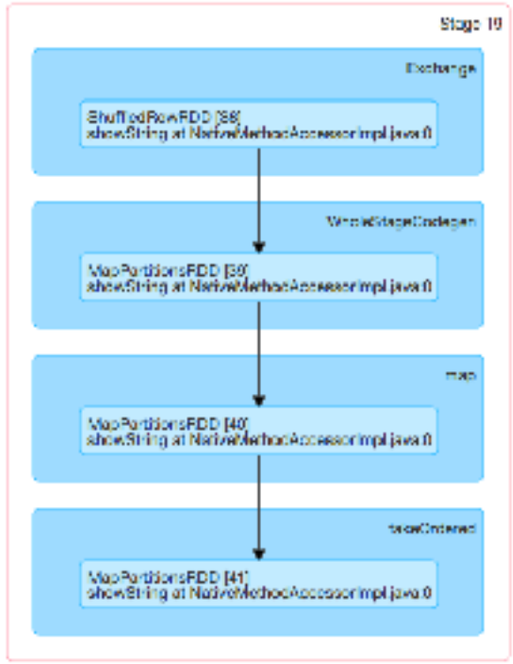
localhost:18080/history/application_1490465719498_0005/stages/stage/?id=19&attempt=0

Spark 2.1.0 | Jobs | Stages | Storage | Environment | Executors | SQL | PySparkShell application UI

Details for Stage 19 (Attempt 0)

Total Time Across All Tasks: 0.4 s
Locality Level Summary: Node local: 12; Process local: 188
Shuffle Read: 8.5 KB / 127

▾ DAG Visualization



▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 200 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0 ms	1 ms	1 ms	1 ms	24 ms
GC Time	0 ms	0 ms	0 ms	0 ms	6 ms
Shuffle Read Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	0.0 B / 0	3.0 KB / 51

▾ Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Shuffle Read Size / Records
2	stdout stderr ip-172-31-39-112.ec2.internal:33243	2 s	200	0	0	200	8.5 KB / 127

Tasks (200)

Page: 1 2 > | 2 Pages. Jump to 1 . Show 100 items in a page. Go

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records	Errors
0	537	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal stdout stderr	2017/03/25 23:06:04	14 ms		0.0 B / 0	

PySparkShell - Details for Stage

localhost:18080/history/application_1490465719498_0005/stages/stage/?id=19&attempt=0

Shuffle Read Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	0.0 B / 0	8.0 KB / 51

- Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Shuffle Read Size / Records
2	stdout stderr ip-172-31-39-112.ec2.internal:33243	2 s	200	0	0	200	8.5 KB / 127

Tasks (200)

Page: 1 2 >

2 Pages. Jump to 1 . Show 100 items in a page. Go

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host		Launch Time	Duration	GC Time	Shuffle Read Size / Records	Errors
0	537	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	14 ms		0.0 B / 0	
1	538	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	5 ms		0.0 B / 0	
2	525	0	SUCCESS	NODE_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	16 ms		511.0 B / 7	
3	539	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	9 ms		0.0 B / 0	
4	540	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
5	541	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
6	542	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
7	543	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
8	544	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
9	545	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	2 ms		0.0 B / 0	
10	546	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
11	547	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
12	548	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	0 ms		0.0 B / 0	
13	549	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
14	550	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
15	551	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
16	552	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	2 ms		0.0 B / 0	
17	553	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	
18	554	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	2 ms		0.0 B / 0	
19	555	0	SUCCESS	PROCESS_LOCAL	2 / ip-172-31-39-112.ec2.internal	stdout stderr	2017/03/25 23:06:04	1 ms		0.0 B / 0	

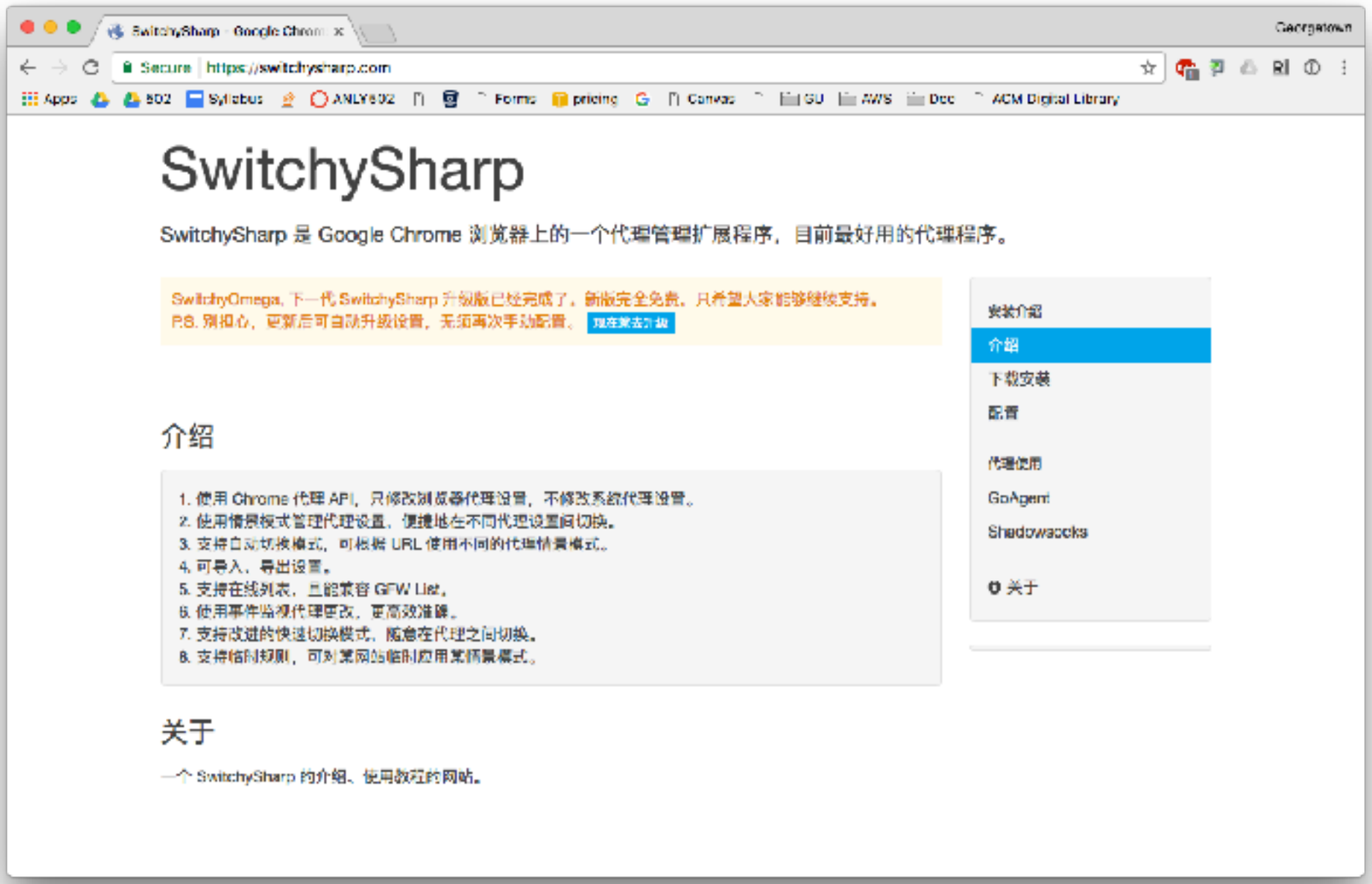
Note: Access to stdout & stderr requires foxyproxy

Proxy options

http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-connect-master-node-proxy.html

FoxyProxy — I can't get it to work anymore

SwitchySharp — ?



JSON

What it is
Demo
Lab

JSON — JavaScript Object Notation
Data Interchange format

JSON — Compact representation format for hierarchical data

Basic data types:

- Number `1, 2, 3`
- String `"this is a string"`, "العَرَبِيَّة" , "汉语; 漢語" (Unicode string)
- Boolean `true, false`
- Array [ ], [1, 2, 3] (Any type may be in an array)
- Objects {"age":24, "height":66} (Also called dictionaries)
- `null`

JSON specified by RFC 7159 and ECMA-404.

Implementations for reading & writing in most languages.

- python: import json

Limitations:

- Numbers: No scientific notation, `Infinity`, `NaN`
- Can only represent a "tree" of objects; no loops.

Some sample JSON

Some students:

```
students = ["Alice","Bob","Charlie","Debra"]
```

Student enrollments

```
students = ["Alice","Bob","Charlie","Debra"]
```

Alice is enrolled in ANLY502 and ANLY503

Bob is enrolled in ANLY502

Charlie is enrolled in ANLY502

Debra is not enrolled in any course.

```
In [1]: students = {}

In [2]: students["Alice"] = {"courses":["ANLY502","ANLY503"]}

In [3]: students["Bob"] = {"courses":["ANLY502"]}

In [4]: students["Charlie"] = {"courses":["ANLY502"]}

In [5]: students["Debra"] = {"courses":[]}
```

Advantages of JSON: It's a storage format that works like code

```
In [7]: students
Out[7]:
{'Alice': {'courses': ['ANLY502', 'ANLY503']},
 'Bob': {'courses': ['ANLY502']},
 'Charlie': {'courses': ['ANLY502']},
 'Debra': {'courses': []}}

In [8]: import json

In [9]: json.dumps(students)
Out[9]: '{"Alice": {"courses": ["ANLY502", "ANLY503"]}, "Bob":
{"courses": ["ANLY502"]}, "Charlie": {"courses": ["ANLY502"]},
"Debra": {"courses": []}}'

In [10]: json.dump(students,fp=open("students.json","w"))

In [12]: json.load(fp=open("students.json","r"))
Out[15]:
{'Alice': {'courses': ['ANLY502', 'ANLY503']},
 'Bob': {'courses': ['ANLY502']},
 'Charlie': {'courses': ['ANLY502']},
 'Debra': {'courses': []}}
```

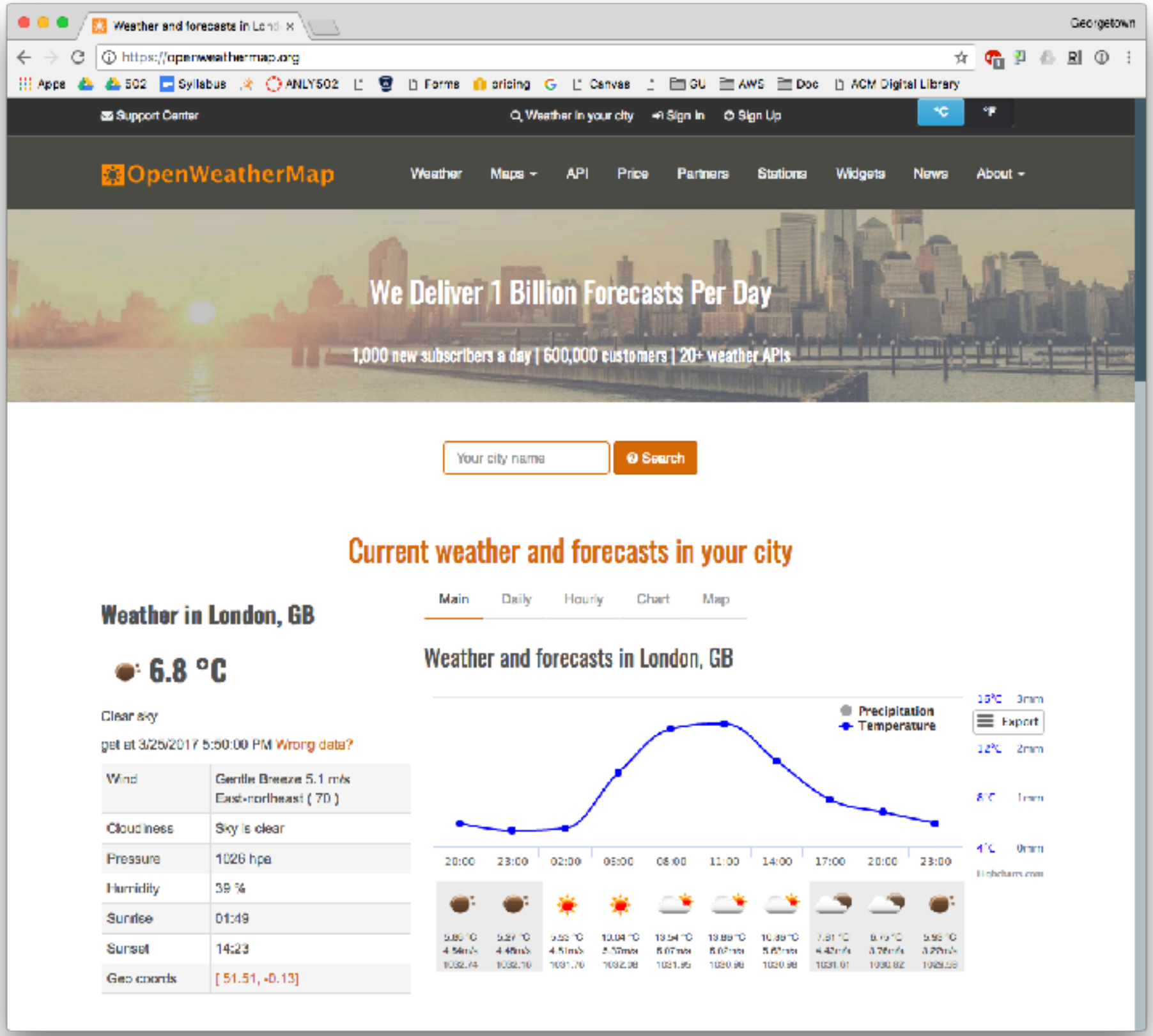
Advantages of JSON: it's easy to extend

```
In [16]: students["Alice"]["email"] = "alice@georgetown.edu"

In [17]: students
Out[17]:
{'Alice': {'courses': ['ANLY502', 'ANLY503'],
           'email': 'alice@georgetown.edu'},
 'Bob': {'courses': ['ANLY502']},
 'Charlie': {'courses': ['ANLY502']},
 'Debra': {'courses': []}}

In [18]:
```

JSON compared to other data interchange formats

	Record-based?	Hierarchical?	Compact?	Extensible
JSON		✔	✔	✔
XML		✔		✔
CSV / TSV	✔			
Google Protocol buffers		✔	✔	
Apache Parquet	✔		✔	
HDF5	✔	✔	✔	✔

JSON is commonly used for web APIs



Weather API- OpenWeatherMa

Georgetown

Secure https://openweathermap.org/api

Apps 502 Syllabus ANLY502 Forms pricing Canvas GU AWS Doc ACM Digital Library

Support Center | Weather in your city | Sign In | Sign Up | °C | °F

OpenWeatherMap

Weather | Maps | API | Price | Partners | Stations | Widgets | News | About

Our weather API is simple, clear and free. We also offer higher levels of support, please see our paid plan options. To access the API you need to sign up for an API key if you are on a free or paid plan.

Current weather data

API doc | Subscribe

- Access current weather data for any location including over 200,000 cities
- Current weather is frequently updated based on global models and data from more than 40,000 weather stations
- Data is available in JSON, XML, or HTML format
- Available for Free and all other paid accounts

5 day / 3 hour forecast

API doc | Subscribe

- 5 day forecast is available at any location or city
- 5 day forecast includes weather data every 3 hours
- Forecast is available in JSON and XML
- Available for Free and all other paid accounts

16 day / daily forecast

API doc | Subscribe

- 16 day forecast is available at any location or city
- 16 day forecast includes daily weather
- Forecast is available in JSON and XML
- Available for Developer, Professional and Enterprise accounts

Historical data

API doc | Subscribe

- Through our API we provide city historical weather data for 37,000+ cities
- Historical data is available for 1 month previous in Starter account, for 1 year previous in Medium accounts, and for 5 years in History Bulk

UV Index BETA

API doc | Subscribe

- Current UV index (Clear Sky) and historical data are available for any geo location (lat/lon)
- Interpreting of the UV Index and recommended protection are provided
- Data is available in JSON
- Available for Professional and Enterprise accounts

Weather map layers

API doc | Subscribe

- Weather maps include precipitation, clouds, pressure, temperature, wind, and more
- Connect our weather maps to your mobile applications and websites
- Use as layers in Direct Tiles, OpenLayers, Leaflet, and Google Maps
- Available for Free and all other paid accounts

Weather stations NEW!

API doc | Subscribe

- API for managing your personal weather stations
- Creation of the station and measurements transfer
- Receiving the aggregated

Bulk downloading

API doc | Subscribe

- We provide number of bulk files with current weather and forecasts
- Current weather bulk is available for 20,000 cities
- Variety of hourly and daily forecast bulks

Air pollution BETA

API doc | Subscribe

- We provide current and historical (since November 2015) data on air pollution with main indexes of CO, O3, NO2 and SO2.
- Air pollution: ways to forecast and

16 day weather forecast- Ope

Secure https://openweathermap.org/forecast16

Support Center | Weather in your city | Sign In | Sign Up | °C | °F

OpenWeatherMap | Weather | Maps | API | Price | Partners | Stations | Widgets | News | About

16 day forecasts is available at any location or city. Forecasts include daily weather and available in JSON or XML format.

Call 16 day / daily forecast data

> Please remember that all Examples of API calls that listed on this page are just samples and do not have any connection to the real API service!

By city name

Description:

You can search 16 day weather forecast with daily average parameters by city name. All weather data can be obtained in JSON and XML formats.

> There is a possibility to receive a central district of the city/town with its own parameters (geographic coordinates/id/name) in API response. Example

API call:

api.openweathermap.org/data/2.5/forecast/daily?q={city name},{country code}&cnt={cnt}

Parameters:

q city name and country code divided by comma, use ISO 3166 country codes

cnt number of days returned (from 1 to 16)

Examples of API calls:

Call 7 days forecast by city name in XML format and metric units
api.openweathermap.org/data/2.5/forecast/daily?q=London&mode=xml&units=metric&cnt=7

By city ID

Description:

You can search 16 day weather forecast with daily average parameters by city ID. API responds with exact result. All weather data can be obtained in JSON and XML format.

List of city ID city.list.json.gz can be downloaded here https://bulk.openweathermap.org/sample/

OpenWeatherMap JSON API format


```
{"cod":"200","message":0.0032,
"city":{"id":1851632,"name":"Shuzenji",
"coord":{"lon":138.933334,"lat":34.966671},
"country":"JP"},
"cnt":10,
"list":[{
    "dt":1406080800,
    "temp":{
        "day":297.77,
        "min":293.52,
        "max":297.77,
        "night":293.52,
        "eve":297.77,
        "morn":297.77},
    "pressure":925.04,
    "humidity":76,
    "weather":[{"id":803,"main":"Clouds","description":"broken clouds","icon"
        ]}
```


Parameters:

- `city`
  - `city.id` City ID
  - `city.name` City name
  - `city.coord`
    - `city.coord.lat` City geo location, latitude
    - `city.coord.lon` City geo location, longitude
  - `city.country` Country code (GB, JP etc.)
- `cod` Internal parameter
- `message` Internal parameter
- `cnt` Number of lines returned by this API call
- `list`
  - `list.dt` Time of data forecasted
  - `list.temp`
    - `list.temp.day` Day temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - `list.temp.min` Min daily temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - `list.temp.max` Max daily temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - `list.temp.night` Night temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - `list.temp.eve` Evening temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - `list.temp.morn` Morning temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.

JSON is more compact than XML

```
<weatherdata>
<location>
 <name>London</name>
 <type/>
 <country>GB</country>
 <timezone/>
 <location altitude="0" latitude="51.50853" longitude="-0.12574" geobase="
</location>
<credit/>
<meta>
 <lastupdate/>
 <calctime>0.0091</calctime>
 <nextupdate/>
</meta>
<sun rise="2015-06-04T03:46:26" set="2015-06-04T20:11:17"/>
<forecast>
 <time day="2015-06-04">
 <symbol number="802" name="scattered clouds" var="03d"/>
 <precipitation/>
 <windDirection deg="148" code="SSE" name="South-southeast"/>
 <windSpeed mps="5.12" name="Gentle Breeze"/>
 <temperature day="23.65" min="17.27" max="23.74" night="17.27" eve="22.9
 <pressure unit="hPa" value="1032.24"/>
 <humidity value="70" unit="%"/>
 <clouds value="scattered clouds" all="36" unit="%"/>
 </time>
</forecast>
</weatherdata>
```

JSON and Spark

Spark can read a JSON dataset as a DataFrame (or Dataset)

Option 1 — Read the JSON on the Driver and "parallelize"

- Reading of JSON is single-threaded; may be slow.
- Easy for small datasets.

Option 2 — Store the JSON into S3 / HDFS and load it.

- Better for large datasets
- Great if the dataset is changing over time (ie: more data being stored in bucket)

Demo: parallelize

```
In [1]: data = []

In [2]: data.append({"color":"red", "height":150})

In [3]: data.append({"color":"blue", "height":125})

In [4]: data.append({"color":"green", "height":75})

In [5]: df = spark.read.json( sc.parallelize( data ))

In [6]: df.show()
+-----+------+
|color|height|
+-----+------+
|  red|   150|
| blue|   125|
|green|    75|
+-----+------+


In [7]: df.select("height").show()
+------+
|height|
+------+
|   150|
|   125|
|    75|
+------+
```

```
In [8]: df.createOrReplaceTempView("data")

In [9]: spark.sql("select color,height from data "
                  "if height>100").show()
```

whoops...

```
In [8]: df.createOrReplaceTempView("data")

In [9]: spark.sql("select color,height from data if height>100").show()
---------------------------------------------------------------------------
Py4JJavaError                             Traceback (most recent call last)
/usr/lib/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
     62         try:
---> 63             return f(*a, **kw)
     64         except py4j.protocol.Py4JJavaError as e:

/usr/lib/spark/python/lib/py4j-0.10.4-src.zip/py4j/protocol.py in get_return_value(answer, gateway_client, target_id, name)
    318                     "An error occurred while calling {0}{1}{2}.\n".
--> 319                     format(target_id, ".", name), value)
    320             else:

Py4JJavaError: An error occurred while calling o42.sql.
: org.apache.spark.sql.catalyst.parser.ParseException:
mismatched input 'height' expecting {<EOF>, ',', 'WHERE', 'GROUP', 'ORDER', 'HAVING', 'LIMIT', 'JOIN', 'CROSS', 'INNER', 'LEFT', 'RIGHT', 'FULL', 'NATURAL', 'LATERAL',
'WINDOW', 'UNION', 'EXCEPT', 'MINUS', 'INTERSECT', 'SORT', 'CLUSTER', 'DISTRIBUTE', 'ANTI'}(line 1, pos 33)

== SQL ==
select color,height from data if height>100
---------------------------------^^^

       at org.apache.spark.sql.catalyst.parser.ParseException.withCommand(ParseDriver.scala:197)
       at org.apache.spark.sql.catalyst.parser.AbstractSqlParser.parse(ParseDriver.scala:99)
       at org.apache.spark.sql.execution.SparkSqlParser.parse(SparkSqlParser.scala:45)
       at org.apache.spark.sql.catalyst.parser.AbstractSqlParser.parsePlan(ParseDriver.scala:53)
       at org.apache.spark.sql.SparkSession.sql(SparkSession.scala:592)
       at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
       at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
       at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
       at java.lang.reflect.Method.invoke(Method.java:498)
       at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
       at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
       at py4j.Gateway.invoke(Gateway.java:280)
       at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
       at py4j.commands.CallCommand.execute(CallCommand.java:79)
       at py4j.GatewayConnection.run(GatewayConnection.java:214)
       at java.lang.Thread.run(Thread.java:745)


During handling of the above exception, another exception occurred:

ParseException                            Traceback (most recent call last)
<ipython-input-9-0a2f7c5ec795> in <module>()
----> 1 spark.sql("select color,height from data if height>100").show()

/usr/lib/spark/python/pyspark/sql/session.py in sql(self, sqlQuery)
    539         [Row(f1=1, f2=u'row1'), Row(f1=2, f2=u'row2'), Row(f1=3, f2=u'row3')]
    540         """
--> 541         return DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped)
    542
    543     @since(2.0)

/usr/lib/spark/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py in __call__(self, *args)
   1131         answer = self.gateway_client.send_command(command)
   1132         return_value = get_return_value(
-> 1133             answer, self.gateway_client, self.target_id, self.name)
   1134
   1135         for temp_arg in temp_args:

/usr/lib/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
     71                 raise AnalysisException(s.split(': ', 1)[1], stackTrace)
```

Relevant part of the error:

```
In [8]: df.createOrReplaceTempView("data")

In [9]: spark.sql("select color,height from data if
height>100").show()


Py4JJavaError: An error occurred while calling o42.sql.
: org.apache.spark.sql.catalyst.parser.ParseException:
mismatched input 'height' expecting {<EOF>, ',', 'WHERE', 'GROUP',
'ORDER', 'HAVING', 'LIMIT', 'JOIN', 'CROSS', 'INNER', 'LEFT', 'RIGHT',
'FULL', 'NATURAL', 'LATERAL', 'WINDOW', 'UNION', 'EXCEPT', 'MINUS',
'INTERSECT', 'SORT', 'CLUSTER', 'DISTRIBUTE', 'ANTI'}(line 1, pos 33)

== SQL ==
select color,height from data if height>100
---------------------------------^^^

ParseException: "\nmismatched input 'height' expecting {<EOF>, ',',
'WHERE', 'GROUP', 'ORDER', 'HAVING', 'LIMIT', 'JOIN', 'CROSS',
'INNER', 'LEFT', 'RIGHT', 'FULL', 'NATURAL', 'LATERAL', 'WINDOW',
'UNION', 'EXCEPT', 'MINUS', 'INTERSECT', 'SORT', 'CLUSTER',
'DISTRIBUTE', 'ANTI'}(line 1, pos 33)\n\n== SQL ==\nselect
color,height from data if
height>100\n---------------------------------^^^\n"
```

Corrected query.

```
In [10]: spark.sql("select color,height from data "
                   "where height>100").show()
+-----+------+
|color|height|
+-----+------+
|  red|   150|
| blue|   125|
+-----+------+


In [11]: df.printSchema()
root
 |-- color: string (nullable = true)
 |-- height: long (nullable = true)


In [12]:
```

Hierarchical JSON can be a bit more challenging

```
In [26]: df2 = spark.read.json( sc.parallelize(
 [ {"father":"fred",
    "children": [{"name":"sam", "age":10},
                 {"name":"betty", "age": 20 }]
   }
 ] ) )


In [29]: df2.printSchema()
root
 |-- children: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- age: long (nullable = true)
 |    |    |-- name: string (nullable = true)
 |-- father: string (nullable = true)


In [30]:
```

```
In [30]: df2.createOrReplaceTempView("families")

In [31]: spark.sql("select * from families").show()
+---------------------+------+
|            children|father|
+---------------------+------+
|[[10,sam], [20,be...|  fred|
+---------------------+------+


In [32]: spark.sql("select father,children.name from families").show()
+------+-------------+
|father|         name|
+------+-------------+
|  fred|[sam, betty]|
+------+-------------+


In [33]: spark.sql("select father,children.age from families").show()
+------+---------+
|father|     age|
+------+---------+
|  fred|[10, 20]|
+------+---------+
```

Lab: catalog.data.gov/datasets?res_format=JSON

Datasets - Data.gov

Secure https://catalog.data.gov/dataset?res_format=JSON

Search Data.Gov

DATA.GOV

DATA TOPICS IMPACT APPLICATIONS DEVELOPERS CONTACT

DATA CATALOG / Datasets Organizations

Search datasets...

Order by: Popular

Datasets ordered by Popular

Formats: JSON ×

Filter by location Clear

Enter location...

Map tiles & Data by OpenStreetMap, under CC BY SA.

Topics Clear All

Local Government (7061)

AAPI (35)

Climate (6)

Agriculture (4)

Education (4)

12,437 datasets found

Consumer Complaint Database 269 recent views

Consumer Financial Protection Bureau — These are complaints we've received about financial products and services.

CSV CSV JSON XML API

Federal

Demographic Statistics By Zip Code 180 recent views

City of New York — Demographic statistics broken down by zipcode

CSV RDF JSON XML

City

Crimes - 2001 to present 170 recent views

City of Chicago — This dataset reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to...

CSV RDF JSON XML

City

U.S. Chronic Disease Indicators (CDI) 159 recent views

U.S. Department of Health & Human Services — CDC's Division of Population Health provides cross-cutting set of 124 indicators that were developed by consensus and that allows states and territories and large...

Federal

Working with JSON

There are two ways to represent a JSON dataset:

- One line per record, each line a JSON value.
- One huge JSON record *This is what ConsumerFinance uses*

```
from pyspark.sql import Row
from collections import OrderedDict

def convert_to_row(d: dict) -> Row:
    return Row(**OrderedDict(sorted(d.items())))
```

—*Converting a local array of dictionaries:*

```
df = sc.parallelize(array_of_dicts).map(convert_to_row).toDF()
```

—*Converting an S3 list of JSON lines:*

```
df = spark.read.format('json').load('s3://bucket/data.json')
df = spark.read.json('s3://bucket/data.json')
```

—*Reading parquet:*

```
df = spark.read.parquet( URL )
df = spark.read.text()
```

NoSQL databases and Amazon DB

NoSQL Databases — Different from traditional databases

SQL / RDBMS	NoSQL
Stores data in tables.	Stores data in a collection of (JSON) "documents"
Data are *normalized*. Each string is only stored once.	Data are *denormalized*. Srings that are keys are stored in each document.
Vertical scaling with bigger hardware	Horizontal scaling by splitting on partition key
Designed for consistency	Designed for speed and *ad hoc* changes

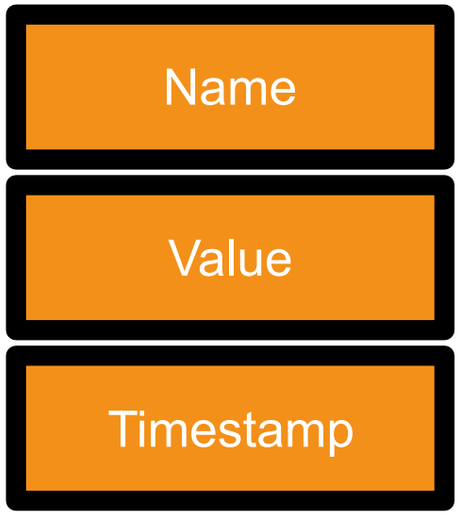
"NoSQL" is a misnomer. They are really "non-relational databases."

Many "NoSQL" databases can now be queries with SQL!

Comparison of NoSQL databases

Kinds of NoSQL database:

- **Column-based** — Accumulo, Cassandra, HBase — data stored tuples consisting of:
  - —*Unique name (references the column)*
  - —*Value (e.g. Ascii, Long, Object)*
  - —*Timestamp — used to determine if the data are valid.*



A "column"
(not RDBMS columns)

- **Document-based** — CouchDB, MongoDB, DynamoDB
  - —*Stores a collection of documents.*
  - —*Some attributes can be indexed.*
  - —*Query language returns documents or part of documents.*

- **Key-value based** — BerkeleyDB, DynamoDB

- **Graph-based** — Neo4J

Most were built for a specific purpose and then generalized.

Interface ≠ Implementation

- The database may appear to store JSON but actually store data in binary structures.

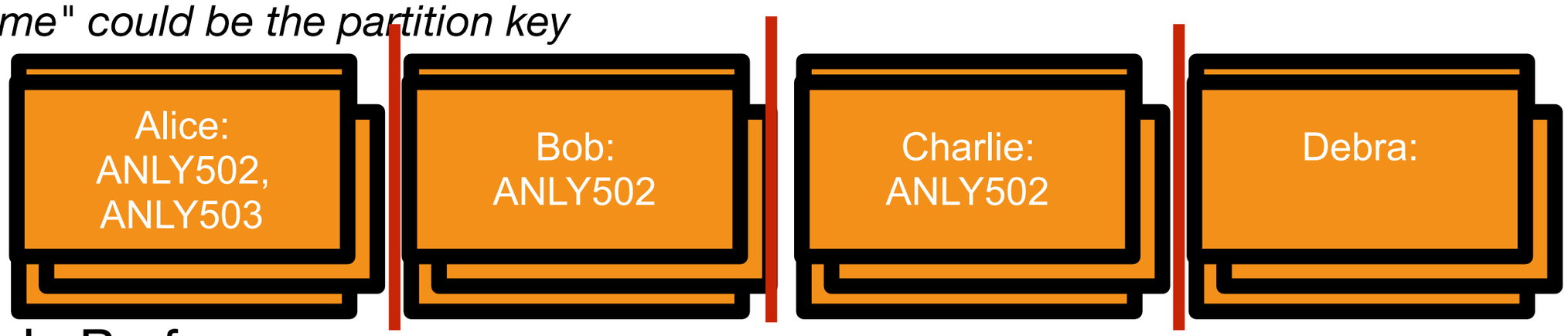
Amazon's DynamoDB

Cloud-hosed NoSQL database service.

—*Originally designed for Amazon's shopping cart.*

- Provides both Key/Value model and Document model.
- Each document must have a "partition key" (like a primary key)
- Database scales by adding nodes and rebalancing along partitions.

— "*Name*" *could be the partition key*



Predictable Performance

- You specify the number of Read & Write units you require.

Limitations:

- Documents limited to 200K
- Indexing with Local Secondary Index (LSI) & Global Secondary Index (GSI) (max 5).
- Not designed as a "data lake" — remove data that you don't need, or be prepared to spend more $$ as the amount of data stored increases.

Using DynamoDB

Create a table!

- Key-Value pairs and Documents are stored in a Table.
- You can have as many tables as you wnat.

Local Version

- You can download a small-scale DynamoDB and run it on your laptop! (It's in Java)

Web GUI

- You can create tables and execute queries

Elastic Map Reduce (EMR)

- Access through Hive Query Language (HiveQL) with a DynamoDB connector
  - *—note: we are using SQL to access a NoSQL database!*
- Copy data between DynamoDB and an S3 bucket or HDFS
- Execute Joins

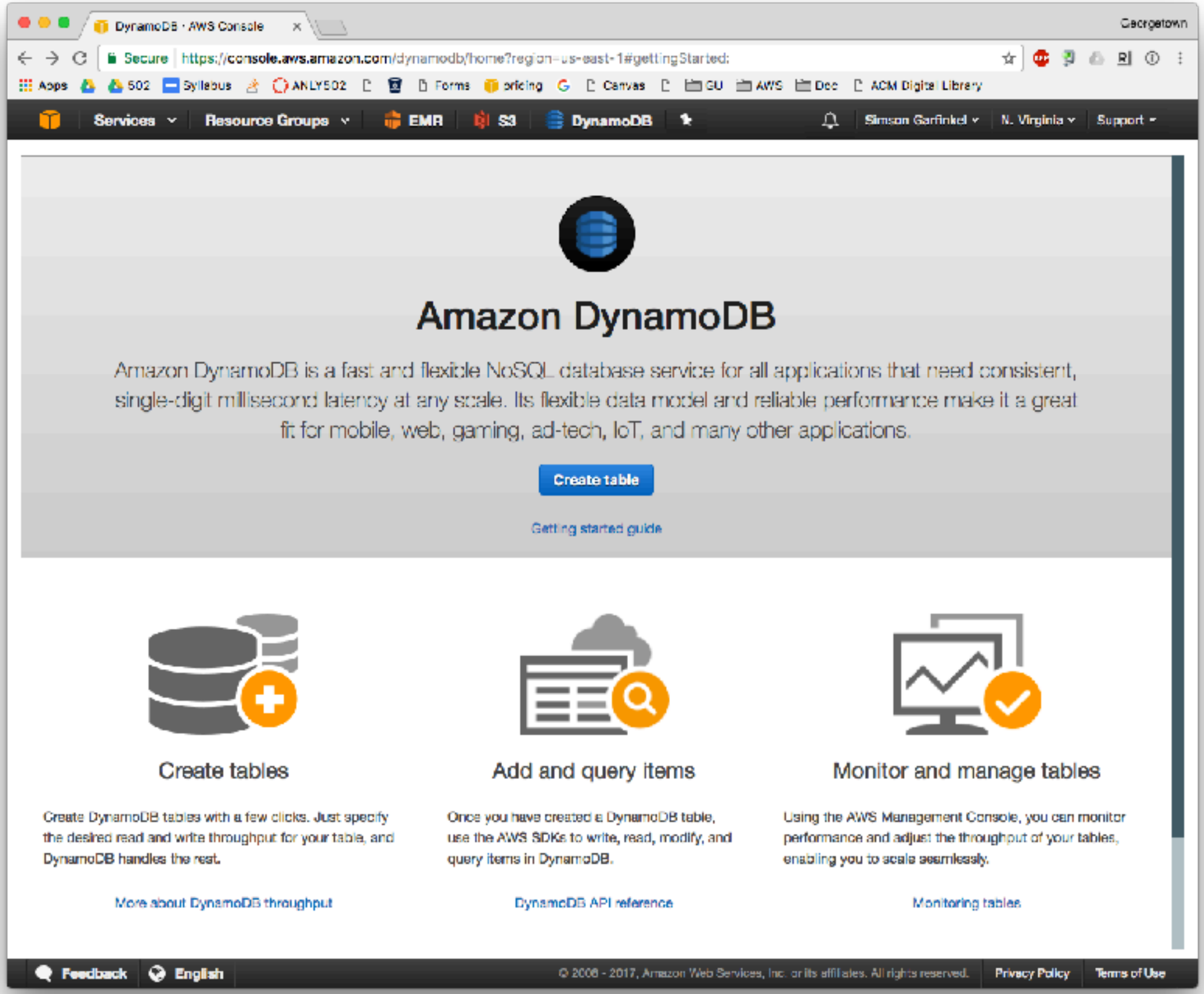
Accessing DynamoDB through HQL

—*http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/EMRforDynamoDB.Tutorial.CopyDataToDDB.html*

```
CREATE EXTERNAL TABLE ddb_features
    (feature_id   BIGINT,
    feature_name  STRING,
    feature_class STRING,
    state_alpha   STRING,
    prim_lat_dec  DOUBLE,
    prim_long_dec DOUBLE,
    elev_in_ft    BIGINT)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES(
    "dynamodb.table.name" = "Features",

    "dynamodb.column.mapping"="feature_id:Id,feature_name:Name,"\
    "feature_class:Class,state_alpha:State,"\
    "prim_lat_dec:Latitude,prim_long_dec:Longitude,"\
    "elev_in_ft:Elevation"
);
```

Submitting this through Hive will require submitting a MapReduce job. (Slow)

Running DynamoDB at Amazon



DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#create-table:

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* students

Primary key* Partition key

name String

☐ Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☑ **Use default settings**

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic 'dynamodb'.

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel | Create

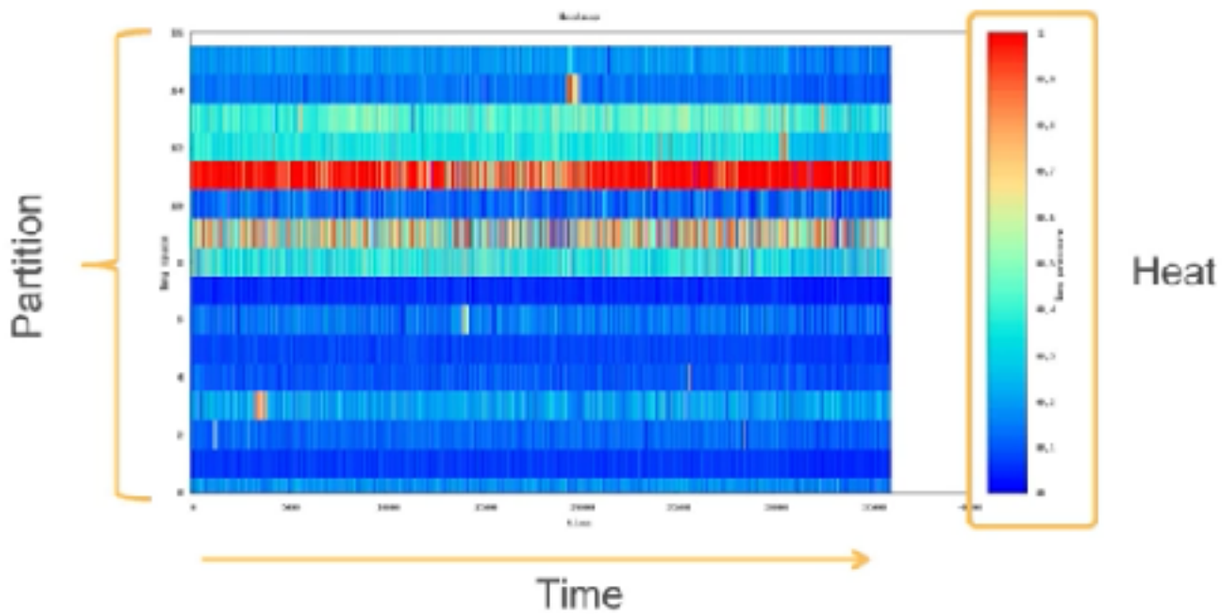
Feedback | English

Privacy Policy | Terms of Use

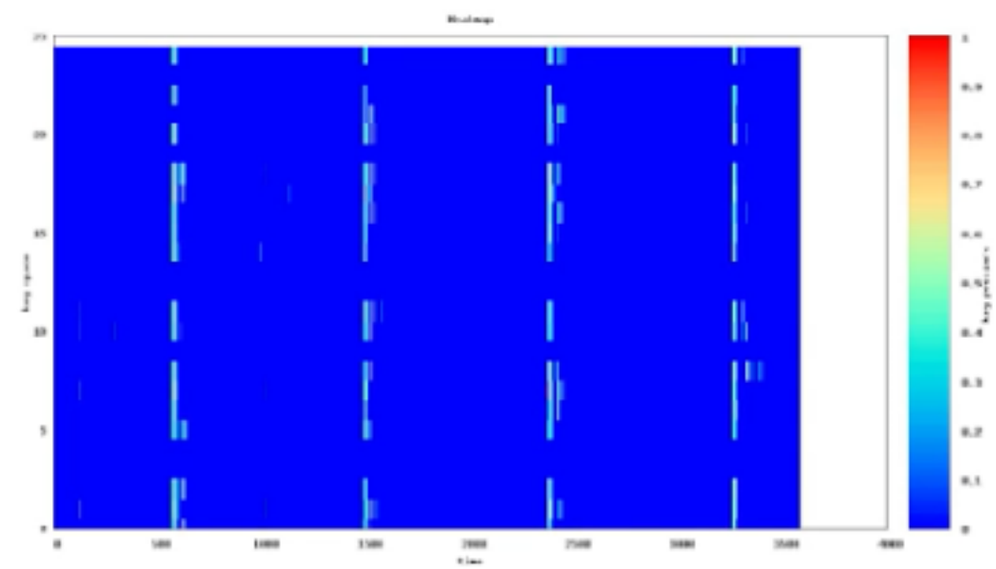
AmazonDB: Choosing a Partition & Sort Key

Partition keys:

- Used to divide work between different partitions (systems)
- Goal — even distribution.



Poorly partitioned



Well partitioned

Choosing a partition key

Goal: random distribution & access based on partition key

Partition key value	Uniformity
User ID, where the application has many users.	Good
Status code, where there are only a few possible status codes.	Bad
Item creation date, rounded to the nearest time period (e.g. day, hour, minute) *Use Sort Key for this.*	Bad
Device ID, where each device accesses data at relatively similar intervals	Good
Device ID, where even if there are a lot of devices being tracked, one is by far more popular than all the others.	Bad

—*http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GuidelinesForTables.html*

Optimizing DynamoDB

Advice from Amazon:

Cache popular items at the application layer

- Amazon ElastiCache, compatible with Redis and Memcached

Use One-to-Many Tables instead of Large Set Attributes

- Store **Forum**, **Thread** and **Reply** in their own tables (rather than all together)
- Shrinks object size
- Decreases the amount of I/O.

Compress large attribute values with gzip or LZO

Store large attribute values in Amazon S3

- DynamoDB gets an S3 URL

DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

DynamoDB

- Dashboard
- Tables
- Reserved capacity

Create table | Actions

Filter by table name

Name
students

students Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Triggers | More

Recent alerts

No CloudWatch alarms have been triggered for this table.

Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

Manage Stream

Table details

Table name	students
Primary partition key	name (String)
Primary sort key	-
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	March 28, 2017 at 11:42:10 AM UTC-4
Provisioned read capacity units	5
Provisioned write capacity units	5
Last decrease time	-
Last increase time	-
Storage size (in bytes)	0 bytes
Item count	0
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:489362128722:table/students

Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

DynamoDB

- Dashboard
- Tables
- Reserved capacity

Create table | Actions

Filter by table name

Name
students

students Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Triggers | More

Create alarm | Edit alarm | Delete alarm

Name	State	Metric
students-ReadCapacityUnitsLimit-BasicAlarm	Insufficient data	ConsumedRead(
students-WriteCapacityUnitsLimit-BasicAlarm	Insufficient data	ConsumedWrite(

Feedback | English

Privacy Policy | Terms of Use

DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

DynamoDB

- Dashboard
- Tables
- Reserved capacity

Create table | Actions

Filter by table name

Name
students

students Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Triggers | More

Provisioned capacity

	Read capacity units	Write capac
Table	5	5

Estimated cost $2.91 / month (Capacity calculator)

Save Cancel

Feedback | English

Privacy Policy | Terms of Use

DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

DynamoDB

- Dashboard
- Tables
- Reserved capacity

Create table | Actions

Filter by table name

Name
students

students Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Triggers | More

Create index | Delete index

Name	Status	Type	Partition key	Sort key

Global Secondary Indexes (GSI) allow you to query efficiently over any field (attribute) in your DynamoDB table. GSIs can treat any table attribute as a key, even attributes not present in all items. More info

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

DynamoDB

- Dashboard
- Tables
- Reserved capacity

Create table | Actions

Filter by table name

Name
students

students Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Triggers | More

Create trigger | Edit/Test trigger | Delete trigger

Function name	State	Last result

DynamoDB triggers connect DynamoDB streams to Lambda functions. Whenever an item in the table is modified, a new stream record is written, which in turn triggers the Lambda function and causes it to execute. More info

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

DynamoDB

- Dashboard
- Tables
- Reserved capacity

Create table | Actions

Filter by table name

Name
students

students Close

Overview | Items | Metrics | Alarms | Capacity | Access control | More

Fine-grained access control is an optional feature that enables additional filtering and control that is helpful for direct database access by mobile apps. Web Identity Federation allows your mobile apps to use identity providers such as **Login with Amazon**, **Facebook**, and **Google**.

Identity provider
- ✓ Facebook
- Google
- Login with Amazon

Action
is

- DeleteItem
- GetItem
- PutItem
- Query
- UpdateItem

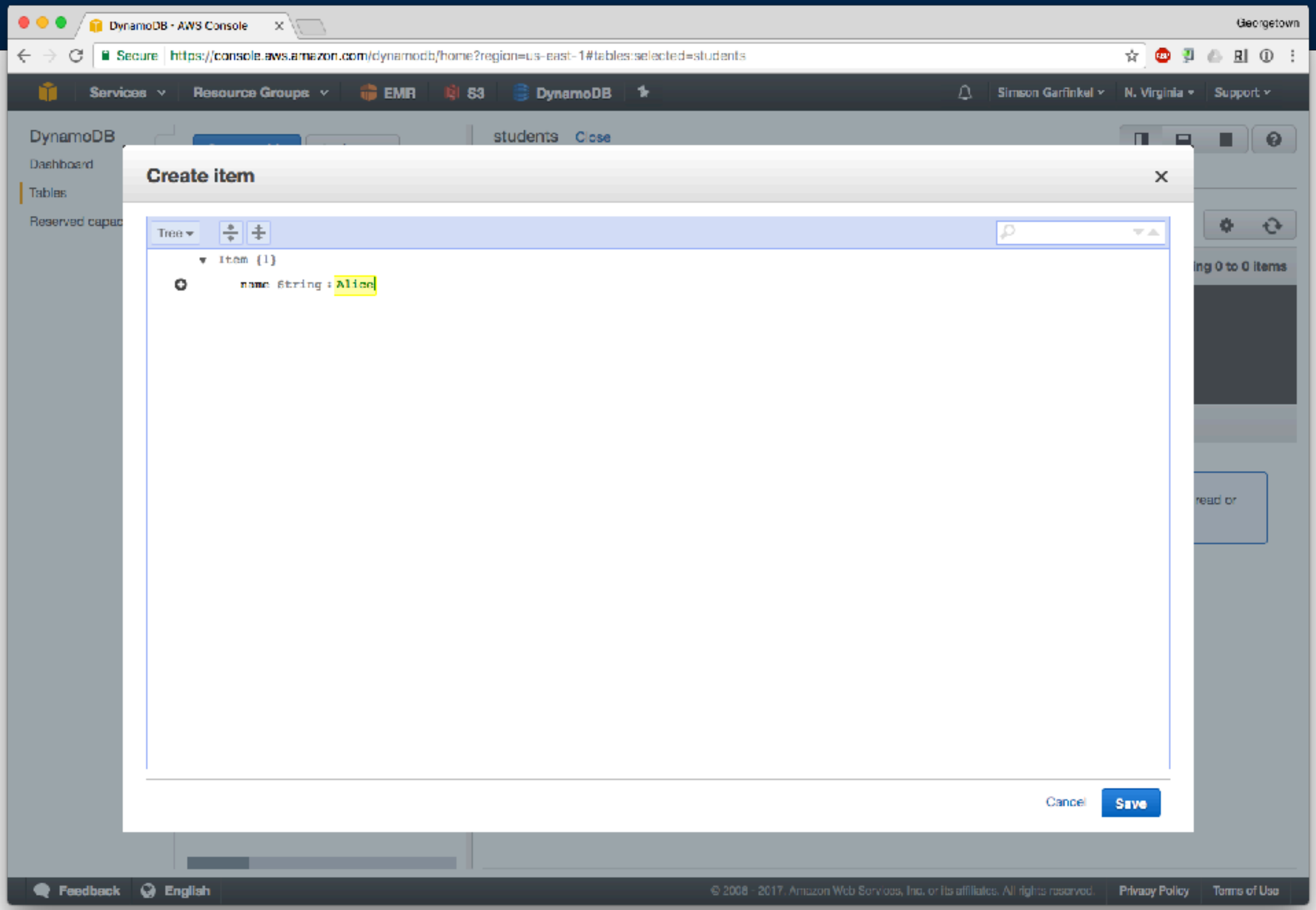
Allowed attributes: All attributes

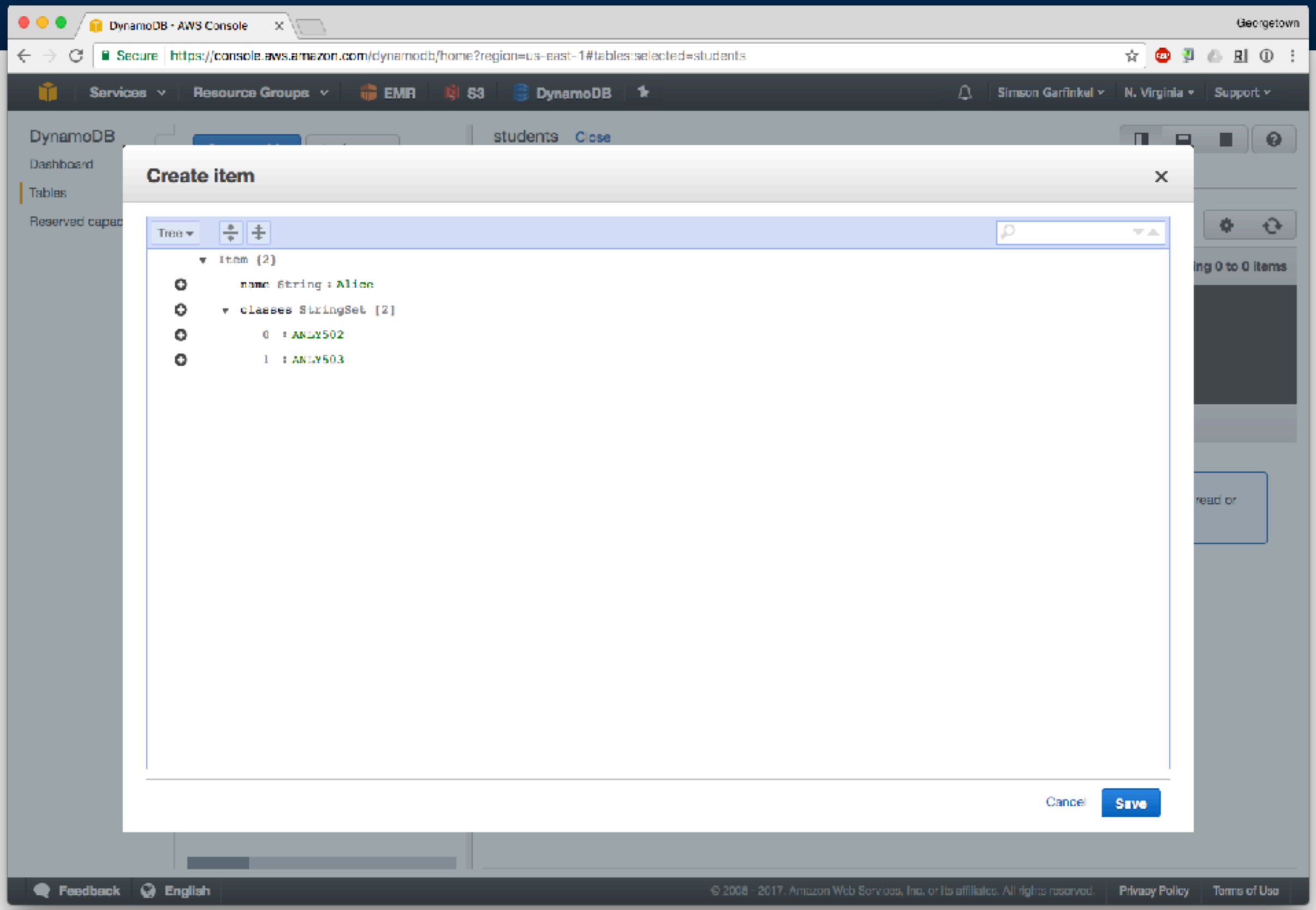
Create policy

```
1 {}
```

Attach policy instructions

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use


DynamoDB - AWS Console
Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students
Services
Resource Groups
EMR
S3
DynamoDB
Simson Garfinkel
N. Virginia
Support
DynamoDB
Dashboard
Tables
students
Create item
Tree
Item {1}
name String : Alice
Cancel
Save
Feedback
English
Privacy Policy
Terms of Use


DynamoDB - AWS Console
Georgetown
Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students
Services
Resource Groups
EMR
S3
DynamoDB
Simson Garfinkel
N. Virginia
Support
DynamoDB
Dashboard
Tables
Reserved capac
students
Close
Create item
Tree
Item {2}
name String : Alice
classes StringSet [2]
0 : ANLY502
1 : ANLY503
Cancel
Save
Feedback
English
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

DynamoDB - AWS Console

Secure https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=students

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | N. Virginia | Support

DynamoDB

- Dashboard
- Tables
- Reserved capacity

Create table | Actions

Filter by table name

Name
students

students Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Triggers | Access control | Tags

Create item | Actions

Scan: [Table] students: name

Viewing 1 to 1 items

Scan | [Table] students: name

Add filter

Start search

name	classes
Alice	{ 'ANLY502', "ANLY503" }

Feedback | English

Privacy Policy | Terms of Use

Accessing DynamoDB from Python — use boto!

```
$ pip install boto3                          # already done on AWS
$ aws configure
```

— *or edit ~/.aws/credentails:*

```
[default]
aws_access_key_id = YOUR_ACCESS_KEY
aws_secret_access_key = YOUR_SECRET_KEY
```

— *http://boto3.readthedocs.io/en/latest/guide/quickstart.html*

```
import boto3
# Let's use Amazon S3
s3 = boto3.resource('s3')

for bucket in s3.buckets.all():
    print(bucket.name)          # Print out bucket names

# Upload a new file
data = open('test.jpg', 'rb')
s3.Bucket('my-bucket').put_object(Key='test.jpg', Body=data)
```

Boto for DynamoDB

—*http://boto3.readthedocs.io/en/latest/guide/dynamodb.html*

```
import boto3

# Get the service resource.
dynamodb = boto3.resource('dynamodb')

table = dynamodb.create_table(TableName=name,
  KeySchema=[], AttributeDefinitions=[],ProvisionedThroughput=[])


table = dynamodb.Table(tableName)
print(table.item_count)
print(table.creation_date_time)
table.put_item(Item={})
response = table.get_item(Key={})
item = response['Item']

table.update_Item(Key={},UpdateExpression=SQL,
                          ExpressionAttributeValues={})

table.delete_item(Key={})
```

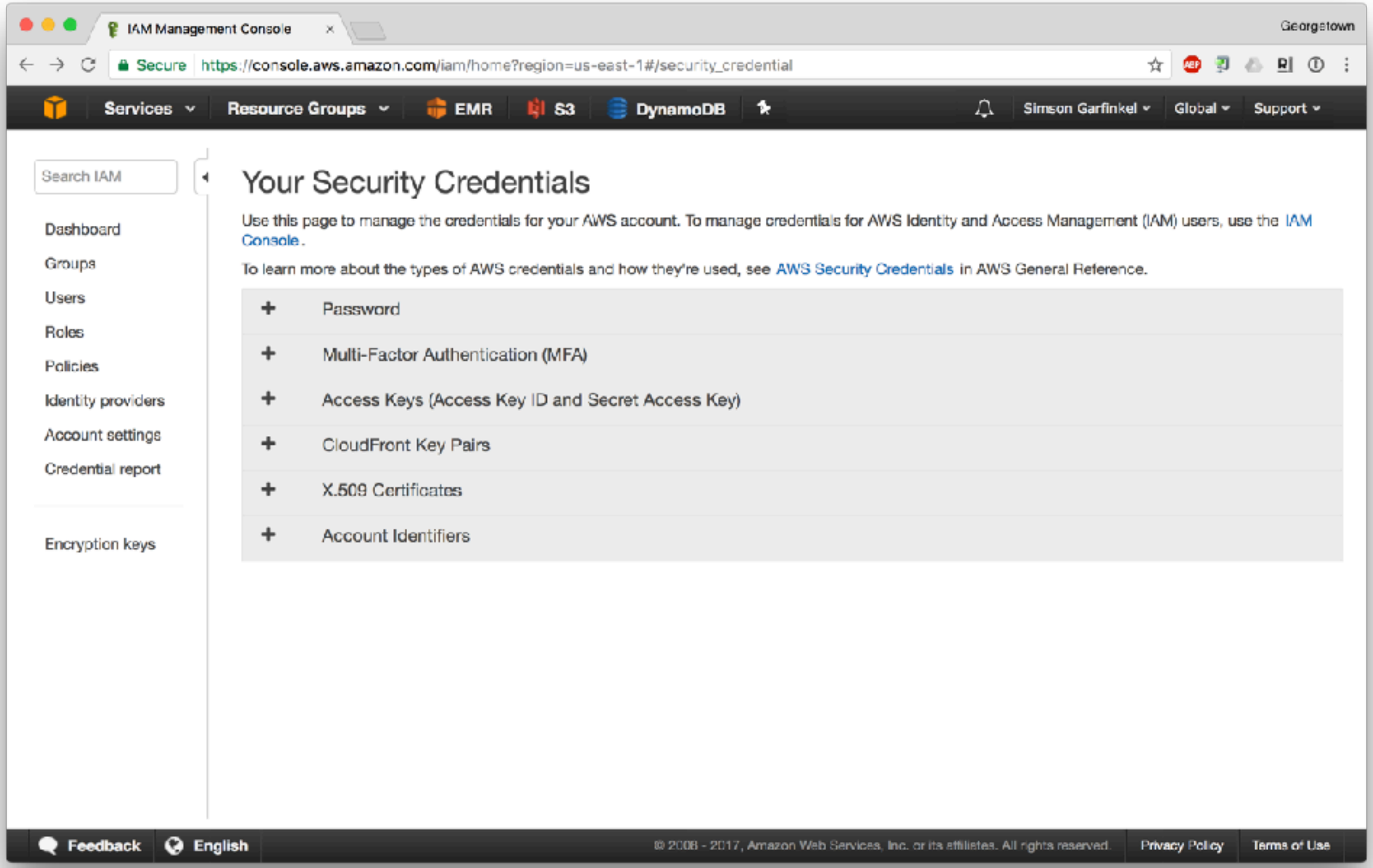
Try it out…

```
In [16]: client = boto3.client('dynamodb',region_name='us-east-1')

In [17]: client.list_tables()
```

1. IPython: Users/simsong (python3.6)

```
ClientError                               Traceback (most recent call last)
<ipython-input-17-c6e8de18887e> in <module>()
----> 1 client.list_tables()

/Users/simsong/anaconda/lib/python3.6/site-packages/botocore/client.py in _api_call(self, *
args, **kwargs)
    251                     "%s() only accepts keyword arguments." % py_operation_name)
    252             # The "self" in this scope is referring to the BaseClient.
--> 253             return self._make_api_call(operation_name, kwargs)
    254
    255         _api_call.__name__ = str(py_operation_name)

/Users/simsong/anaconda/lib/python3.6/site-packages/botocore/client.py in _make_api_call(se
lf, operation_name, api_params)
    541             error_code = parsed_response.get("Error", {}).get("Code")
    542             error_class = self.exceptions.from_code(error_code)
--> 543             raise error_class(parsed_response, operation_name)
    544         else:
    545             return parsed_response

ClientError: An error occurred (AccessDeniedException) when calling the ListTables operatio
n: User: arn:aws:iam::489362128722:user/Dnace is not authorized to perform: dynamodb:ListTa
bles on resource: *

In [18]:
```

Add attribute to IAM user



IAM Management Console

Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/users

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Search IAM

- Dashboard
- Groups
- Users
- Roles
- Policies
- Identity providers
- Account settings
- Credential report

- Encryption keys

Add user | Delete user

Find users by username or access key

Showing 2 res

User name	Groups	Password	Last sign-in	Access keys	Creation time
anly502	1		N/A	1 active	2016-02-14 15:49 EDT
Dnace	1		N/A	1 active	2017-02-19 08:34 EDT

https://console.aws.amazon.com/iam/home?region=us-east-1#/users/Dnace

Privacy Policy | Terms of Use

IAM Management Console

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/users/Dnace

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Search IAM

- Dashboard
- Groups
- Users
- Roles
- Policies
- Identity providers
- Account settings
- Credential report

- Encryption keys

Users > Dnace

Summary

User ARN	arn:aws:iam::489362128722:user/Dnace
Path	/
Creation time	2017-02-19 08:34 EDT

Permissions | **Groups (1)** | **Security credentials** | **Access Advisor**

Add permissions **Attached policies: 2**

Policy name	Policy type
Attached from group	
AmazonEC2FullAccess	AWS managed policy from group mygroup
AmazonS3FullAccess	AWS managed policy from group mygroup

Add inline policy

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

IAM Management Console

Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/users/Dnace$addPermissions?step=permissions

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Add permissions to Dnace

1 Permissions — 2 Review

Grant permissions

Use IAM policies to grant permissions. You can assign an existing policy or create a new one.

- Add user to group
- Copy permissions from existing user
- Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Create group | Refresh

Search

Showing 0 results

Group	Attached policies
No results	

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

IAM Management Console

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/users/Dnace$addPermissions?step=permissions&permissionTyp...

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Add permissions to Dnace

Create group

Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Learn more

Group name DynamoDB_RW

Create policy | Refresh

Filter: Policy type | Search | Showing 252 results

Policy name	Type	Attachments	Description
AdministratorAccess	Job function	0	Provides full access to AWS services and resources.
AmazonAPIGatewayAdm...	AWS managed	0	Provides full access to create/edit/delete APIs in Amazon A...
AmazonAPIGatewayInvo...	AWS managed	0	Provides full access to invoke APIs in Amazon API Gateway.

Cancel | Create group

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/users/Dnace$addPermissions?step=permissions&permissionTyp...

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Use IAM policies to grant permissions. You can assign an existing policy or create a new one.

Add user to group

Copy permissions from existing user

Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Create group | Refresh

Search

Showing 1 result

Group	Attached policies
DynamoDB_RW	None

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

IAM Management Console | IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/groups/DynamoDB_RW

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Search IAM

- Dashboard
- Groups
- Users
- Roles
- Policies
- Identity providers
- Account settings
- Credential report

- Encryption keys

IAM > Groups > DynamoDB_RW

Summary

Group ARN:	arn:aws:iam::489362128722:group/DynamoDB_RW
Users (in this group):	0
Path:	/
Creation Time:	2017-03-28 15:56 EDT

Users | Permissions | Access Advisor

Managed Policies

There are no managed policies attached to this group.

Attach Policy

Inline Policies

Feedback | English | | Privacy Policy | Terms of Use

IAM Management Console | IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/groups/DynamoDB_RW

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Attach Policy

Attach Policy

Select one or more policies to attach. Each group can have up to 10 policies attached.

Filter: Policy Type | Filter | Showing 252 results

Policy Name	Attached Entities	Creation Time	Edited Time
AmazonEC2FullAccess	2	2015-02-06 13:40 EDT	2015-02-06 13:40 EDT
AmazonElasticMapReduce...	2	2015-02-06 13:41 EDT	2016-11-17 18:17 EDT
AmazonS3FullAccess	2	2015-02-06 13:40 EDT	2015-02-06 13:40 EDT
AmazonElasticMapReduce...	1	2016-11-17 20:09 EDT	2016-11-17 20:09 EDT
AmazonElasticMapReduce...	1	2015-02-06 13:41 EDT	2015-05-13 17:27 EDT
AmazonRDSEnhancedMon...	1	2015-11-11 14:58 EDT	2015-11-11 14:58 EDT
AdministratorAccess	0	2015-02-06 13:39 EDT	2015-02-06 13:39 EDT
AmazonAPIGatewayAdmin...	0	2015-07-09 13:34 EDT	2015-07-09 13:34 EDT
AmazonAPIGatewayInvoke...	0	2015-07-09 13:36 EDT	2015-07-09 13:36 EDT
AmazonAPIGatewayPushT...	0	2015-11-11 18:41 EDT	2015-11-11 18:41 EDT
AmazonAppStreamFullAcc...	0	2015-02-06 13:40 EDT	2015-02-06 13:40 EDT

Cancel | Attach Policy

IAM Management Console | IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/groups/DynamoDB_RW

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Attach Policy

Attach Policy

Select one or more policies to attach. Each group can have up to 10 policies attached.

Filter: Policy Type dynamodb

Showing 5 results

	Policy Name	Attached Entities	Creation Time	Edited Time
☑	AmazonDynamoDBFullAcc...	0	2015-02-06 13:40 EDT	2015-11-11 21:17 EDT
☑	AmazonDynamoDBFullAcc...	0	2015-02-06 13:40 EDT	2015-11-11 21:17 EDT
☑	AmazonDynamoDBReadO...	0	2015-02-06 13:40 EDT	2017-02-27 12:59 EDT
☐	AWSLambdaDynamoDBEx...	0	2015-04-09 11:09 EDT	2015-04-09 11:09 EDT
☐	AWSLambdaInvocation-Dy...	0	2015-02-06 13:40 EDT	2015-02-06 13:40 EDT

Cancel | Attach Policy

IAM Management Console | IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/groups/DynamoDB_RW

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Search IAM

- Dashboard
- Groups
- Users
- Roles
- Policies
- Identity providers
- Account settings
- Credential report

- Encryption keys

Summary

Group ARN:	arn:aws:iam::489362128722:group/DynamoDB_RW
Users (in this group):	0
Path:	/
Creation Time:	2017-03-26 15:56 EDT

Users | Permissions | Access Advisor

Managed Policies

The following managed policies are attached to this group. You can attach up to 10 managed policies.

Attach Policy

Policy Name	Actions
AmazonDynamoDBFullAccess	Show Policy | Detach Policy | Simulate Policy
AmazonDynamoDBReadOnlyAccess	Show Policy | Detach Policy | Simulate Policy
AmazonDynamoDBFullAccesswithDataPipeline	Show Policy | Detach Policy | Simulate Policy

Inline Policies

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

IAM Management Console | IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/groups/DynamoDB_RW

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Search IAM

- Dashboard
- Groups
- Users
- Roles
- Policies
- Identity providers
- Account settings
- Credential report
- Encryption keys

IAM > Groups > DynamoDB_RW

Summary

Group ARN:	arn:aws:iam::489362128722:group/DynamoDB_RW
Users (in this group):	0
Path:	/
Creation Time:	2017-03-28 15:56 EDT

Users | Permissions | Access Advisor

This group does **not contain any** users.

Add Users to Group

Feedback | English | | Privacy Policy | Terms of Use

IAM Management Console | IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/groups/DynamoDB_RW

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Add Users to Group

Select users to add to the group **DynamoDB_RW**

Filter

Showing 2 results

	User Name	Groups	Password	Password Last Used	Access Keys	Creation Time
☐	anly502	1		N/A	1 active	2016-02-14 15:49...
☐	Dnace	1		N/A	1 active	2017-02-19 08:34...

Cancel | Add Users

IAM Management Console | IAM Management Console | IAM Management Console | Georgetown

Secure https://console.aws.amazon.com/iam/home?region=us-east-1#/groups/DynamoDB_RW

Services | Resource Groups | EMR | S3 | DynamoDB | Simson Garfinkel | Global | Support

Add Users to Group

Select users to add to the group **DynamoDB_RW**

Filter

Showing 2 results

	User Name	Groups	Password	Password Last Used	Access Keys	Creation Time
☑	anly502	1		N/A	1 active	2016-02-14 15:49...
☑	Dnace	1		N/A	1 active	2017-02-19 08:34...

Cancel | Add Users

1. IPython: Users/simsong (python3.6)

```
/Users/simsong/anaconda/lib/python3.6/site-packages/botocore/client.py in _make_api_call(se
lf, operation_name, api_params)
    541             error_code = parsed_response.get("Error", {}).get("Code")
    542             error_class = self.exceptions.from_code(error_code)
--> 543             raise error_class(parsed_response, operation_name)
    544         else:
    545             return parsed_response

ClientError: An error occurred (AccessDeniedException) when calling the ListTables operatio
n: User: arn:aws:iam::489362128722:user/Dnace is not authorized to perform: dynamodb:ListTa
bles on resource: *

In [18]: client.list_tables()
Out[18]:
{'ResponseMetadata': {'HTTPHeaders': {'content-length': '27',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Sun, 26 Mar 2017 19:59:39 GMT',
   'x-amz-crc32': '399698329',
   'x-amzn-requestid': 'ERT2MCPVJ9BE9GQUKMR65IFV0FVV4KQNSO5AEMVJF66Q9ASUAAJG'},
  'HTTPStatusCode': 200,
  'RequestId': 'ERT2MCPVJ9BE9GQUKMR65IFV0FVV4KQNSO5AEMVJF66Q9ASUAAJG',
  'RetryAttempts': 0},
 'TableNames': ['students']}

In [19]:
```

```
In [23]: client.scan(TableName='students')
Out[23]:
{'Count': 1,
 'Items': [{'classes': {'SS': ['ANLY502', 'ANLY503']},
   'name': {'S': 'Alice'}}],
 'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
   'content-length': '100',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Mon, 27 Mar 2017 00:37:43 GMT',
   'server': 'Server',
   'x-amz-crc32': '215185486',
   'x-amzn-requestid':
'MFR47QP5IJ9METHPU420QTUGHVVV4KQNSO5AEMVJF66Q9ASUAAJG'},
   'HTTPStatusCode': 200,
   'RequestId':
'MFR47QP5IJ9METHPU420QTUGHVVV4KQNSO5AEMVJF66Q9ASUAAJG',
   'RetryAttempts': 0},
 'ScannedCount': 1}

In [24]:
```

```
In [31]: table.put_item(Item={'name':'Bob',"courses":["ANLY502"]})
Out[31]:
{'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
   'content-length': '2',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Mon, 27 Mar 2017 00:43:26 GMT',
   'server': 'Server',
   'x-amz-crc32': '2745614147',
   'x-amzn-requestid': 'UVC56BU385MH9S6HI8H0SV24PNVV4KQNSO5AEMVJF66Q9ASUAAJG'},
  'HTTPStatusCode': 200,
  'RequestId': 'UVC56BU385MH9S6HI8H0SV24PNVV4KQNSO5AEMVJF66Q9ASUAAJG',
  'RetryAttempts': 0}}

In [32]: table.put_item(Item={'name':'Charlie',"courses":["ANLY502"]})
Out[32]:
{'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
   'content-length': '2',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Mon, 27 Mar 2017 00:43:38 GMT',
   'server': 'Server',
   'x-amz-crc32': '2745614147',
   'x-amzn-requestid': 'VOOHJ8G8V3E281K2QH55VDRK23VV4KQNSO5AEMVJF66Q9ASUAAJG'},
  'HTTPStatusCode': 200,
  'RequestId': 'VOOHJ8G8V3E281K2QH55VDRK23VV4KQNSO5AEMVJF66Q9ASUAAJG',
  'RetryAttempts': 0}}

In [33]: table.put_item(Item={'name':'Debra',"courses":[]})
Out[33]:
{'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
   'content-length': '2',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Mon, 27 Mar 2017 00:43:50 GMT',
   'server': 'Server',
   'x-amz-crc32': '2745614147',
   'x-amzn-requestid': '9ANN1CUFQTGJPQELS8EVI6KCDVVV4KQNSO5AEMVJF66Q9ASUAAJG'},
  'HTTPStatusCode': 200,
  'RequestId': '9ANN1CUFQTGJPQELS8EVI6KCDVVV4KQNSO5AEMVJF66Q9ASUAAJG',
  'RetryAttempts': 0}}
```

```
In [34]: table.scan()
Out[34]:
{'Count': 4,
 'Items': [{'courses': [], 'name': 'Debra'},
  {'classes': {'ANLY502', 'ANLY503'}, 'name': 'Alice'},
  {'courses': ['ANLY502'], 'name': 'Charlie'},
  {'courses': ['ANLY502'], 'name': 'Bob'}],
 'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
   'content-length': '256',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Mon, 27 Mar 2017 00:44:05 GMT',
   'server': 'Server',
   'x-amz-crc32': '3326768074',
   'x-amzn-requestid':
'S5FQ6KKUJQS6RKBJ56EQDBHIHBVV4KQNSO5AEMVJF66Q9ASUAAJG'},
  'HTTPStatusCode': 200,
  'RequestId':
'S5FQ6KKUJQS6RKBJ56EQDBHIHBVV4KQNSO5AEMVJF66Q9ASUAAJG',
  'RetryAttempts': 0},
 'ScannedCount': 4}

In [35]:
```

```
In [35]: from boto3.dynamodb.conditions import Key,Attr

In [36]: table.query(KeyConditionExpression=Key('name').eq('Alice'))
Out[36]:
{'Count': 1,
 'Items': [{'classes': {'ANLY502', 'ANLY503'}, 'name': 'Alice'}],
 'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
   'content-length': '100',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Mon, 27 Mar 2017 00:46:07 GMT',
   'server': 'Server',
   'x-amz-crc32': '215185486',
   'x-amzn-requestid':
'G8I3T35PC6MVA27I2A3L1NN7BFVV4KQNSO5AEMVJF66Q9ASUAAJG'},
   'HTTPStatusCode': 200,
   'RequestId': 'G8I3T35PC6MVA27I2A3L1NN7BFVV4KQNSO5AEMVJF66Q9ASUAAJG',
   'RetryAttempts': 0},
 'ScannedCount': 1}

In [37]:
```

More information at:

- *http://docs.aws.amazon.com/amazondynamodb/latest/gettingstartedguide/GettingStarted.Python.html*

https://www.pexels.com/photo/people-apple-iphone-writing-154/

Homework and L10 Preview

Calendar

Tue Mar 28 — Final Project Group Proposals

- Send your group name & members to **anly502@nitroba.org**
- When your group is created on Canvas, upload your proposal!

Mon April 3 — L10: Scalable Machine Learning with Spark (Vaisman)

Mon April 10 — No class

Mon April 10 – Fri April 14 — Final Project Online Clinic

Mon April 17 — No class Easter Break

Mon April 24 — L11: Streaming Databases / Graph Databases (Vaisman)

Mon May 1 — L12: Final Project Presentations

Wed May 10 — Final Projects Due

Mon May 15 — Grades due for graduating students (1 student)

Mon May 22 — Grades due for graduate students

Readings & Watching

Required Readings:

- EMR Best Practices (all 38 pages!)
- Python and DynamoDB

Optional Reading:

- History of DynamoDB:
  - *http://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html*

Required Videos:

- Getting Started with DynamoDB (7 minutes)
- How to Design NoSQL Tables and Avoid Hot Keys (6 minutes)

Optional Videos:

- Getting Started with Amazon DynamoDB, April 2017 (56 minutes)
- Introduction to Managed Database Services on AWS (1 hour, 5 minutes)