OPS 813: Cloud Computing

-Today's plan:

1) Case #3: Stock Market data repository & secondary analysis
2) Dataproc
3) BigQuery
4) Exam Prep/Review

Qwik Labs (HW1-HW3)

1) You should now have three badges. Congratulations! Showcase them on your resume/linkedin.

2) Remember to please follow these instructions: https://support.google.com/qwiklabs/answer/9222527?hl=en&ref_topic=9139328 and email me (ns27) the link to your public profile.

GCP Essentials

Baseline: Data, ML, AI

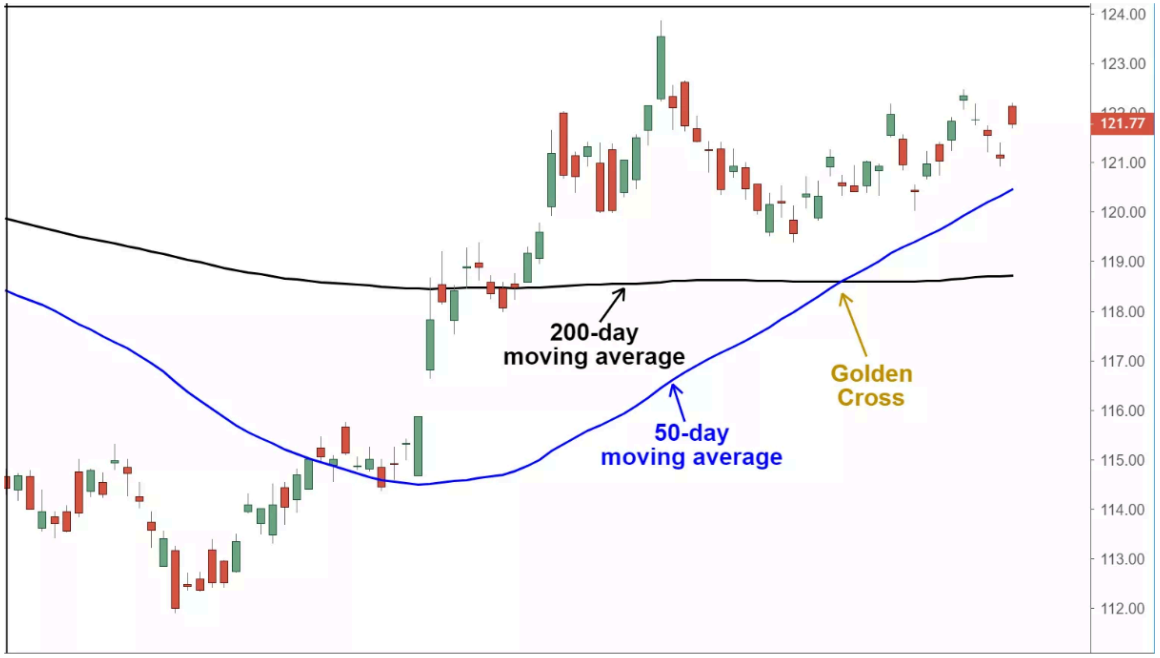
Data Engineering

Case #3

- Use the stock data from this link:
  https://www.kaggle.com/qks1lver/amex-nyse-nasdaq-stock-histories/home
  and Dataproc or some parallelization technology in order to:
- Steps (1) find all ‘golden crosses’, (2) what percentage of those golden crosses result in a price increase by more than 1%, 2%, 3%, 4%, 5%, 10%, 15%, 20% over a week, two weeks, three weeks, a month, two months, three months, four months, five months, six months? (3) find all ‘death crosses’ (when 50DMA crosses below the 200DMA), (4) what percentage of those death crosses result in a price decrease by more than 1%, 2%, 3%, 4%, 5%, 10%, 15%, 20% over a week, two weeks, three weeks, a month, two months, three months, four months, five months, six months?, (5) Does your analytics support the hypothesis of a golden cross or death cross?, (6) as a group choose 6 or 7 stocks that seem interesting to buy and hold for a month.
- Every group will be handing in a report along with their Python code. One group will also present.
- Note that efficiency will be important.

Case #3

‘Golden cross’:



‘Death cross’:

Last time: Hadoop/Map Reduce and Spark

Last time: Spark RDDs

- RDDs are immutable, lazily evaluated, and cacheable
- There are two types of RDD operations:
  - Transformations
  - Actions

Last time: Spark RDD Actions

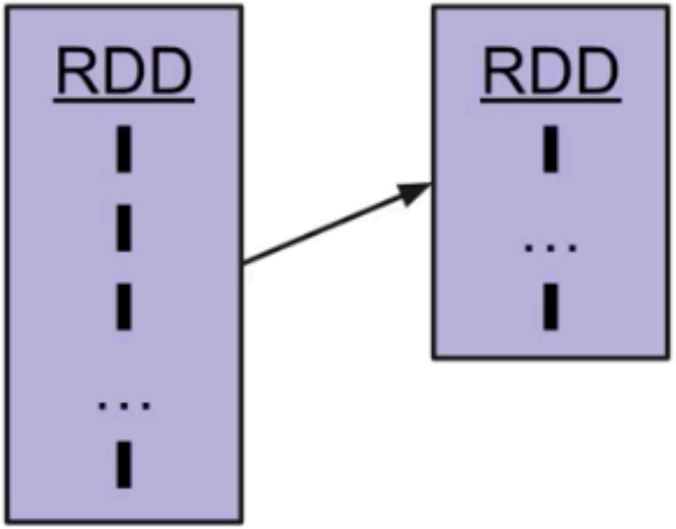
- Collect - Return all the elements of the RDD as an array at the driver program.
- Count - Return the number of elements in the RDD
- First - Return the first element in the RDD
- Take - Return an array with the first n elements of the RDD

Last time: Spark RDD Transformations

- Basic Transformations
  - Filter
  - Map
  - FlatMap

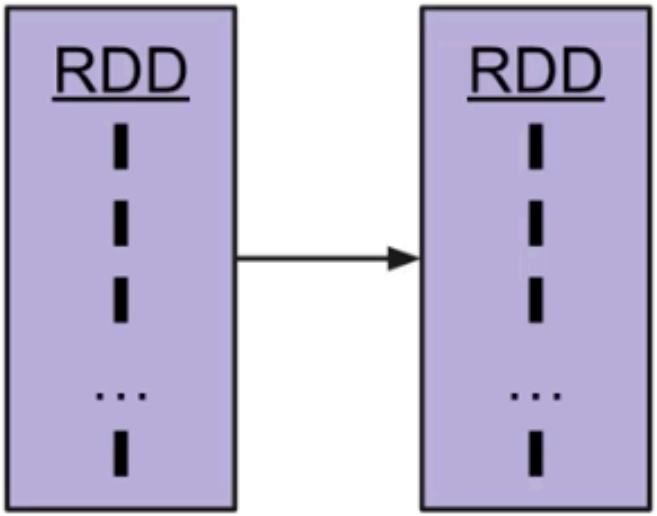
Last time: Spark RDD Transformations

- RDD.filter()
  - Applies a function to each element and returns elements that evaluate to true



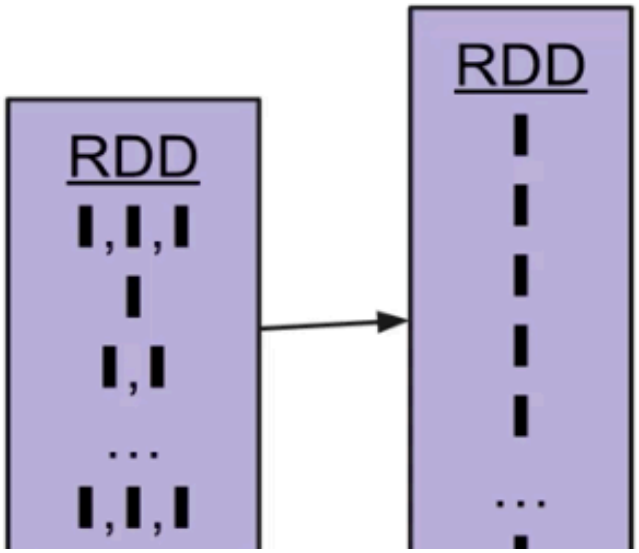
Last time: Spark RDD Transformations

- RDD.map()
  - Transforms each element and preserves # of elements, very similar idea to pandas .apply()



Last time: Spark RDD Transformations

- RDD.flatMap()
  - Transforms each element into 0-N elements and changes # of elements



- Map()
  - Grabbing first letter of a list of names
- FlatMap()
  - Transforming a corpus of text into a list of words

Last time: Pair RDDs

- Often RDDs will be holding their values in tuples
  - (key,value)
- This offers better partitioning of data and leads to functionality based on reduction

Last time: Reduce and ReduceByKey

- Reduce()
  - An action that will aggregate RDD elements using a function that returns a single element
- ReduceByKey()
  - An action that will aggregate Pair RDD elements using a function that returns a Pair RDD
- These ideas are similar to a Group By operation

Last time: Spark rapidly changing

- Spark is being continually developed and new releases come out often!
- The Spark Ecosystem now includes:
  - Spark SQL
  - Spark DataFrames
  - MLlib
  - GraphX
  - Spark Streaming

Last time: Setup Spark

- Let's use the bare metal instance on which you installed Jupyter.
- Need Scala (scala depends on Java):
  sudo apt-get update
  sudo apt-get install default-jre (installs Java)
  java –version
  sudo apt-get install scala
  scala –version
- Need py4j:
  which pip (otherwise need conda install pip)
  pip install py4j
- Install Spark:
  wget https://archive.apache.org/dist/spark/spark-2.3.3/spark-2.3.3-bin-hadoop2.7.tgz
  uncompress: gunzip and tar -xvf

Last time: Setup Spark

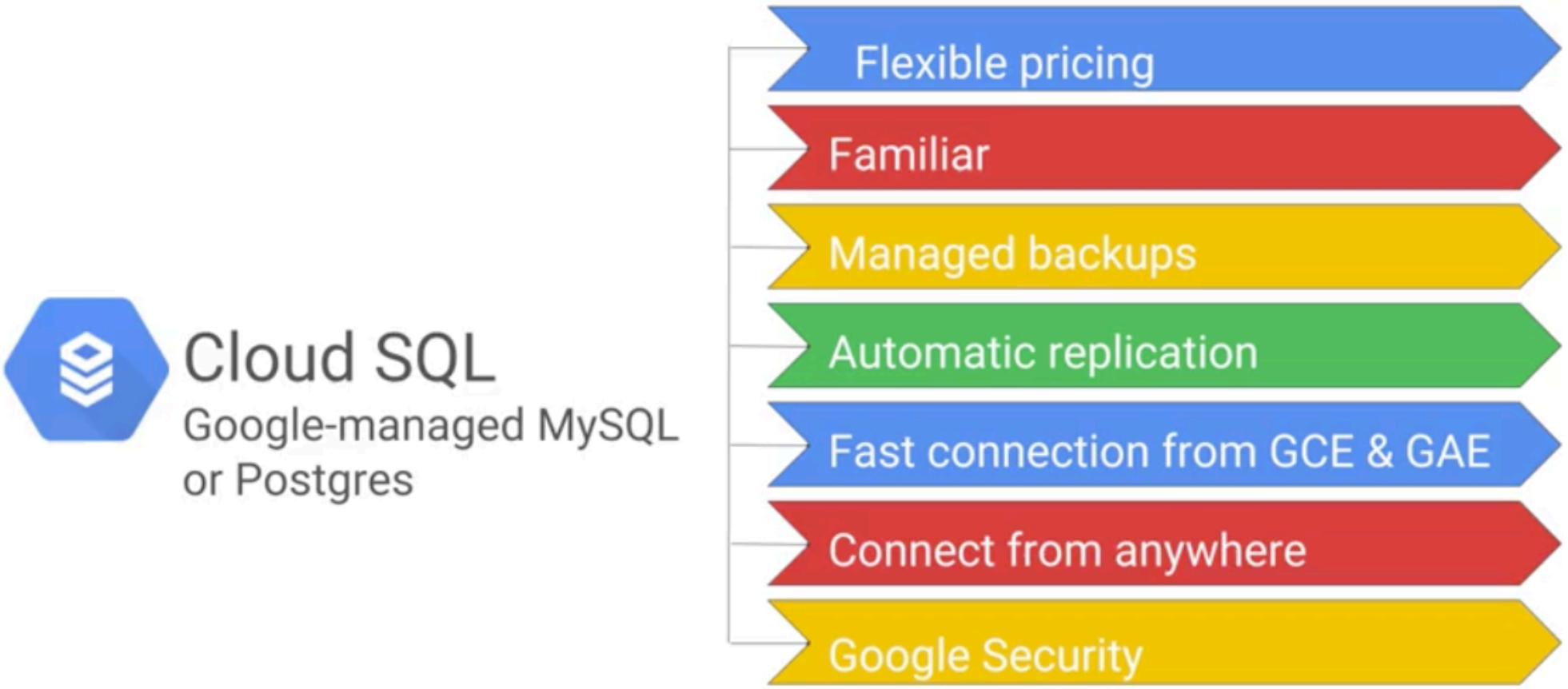
- Tell it where to find Spark:
  export SPARK_HOME='/home/ns27/spark-2.3.3-bin-hadoop2.7'
  export PATH=$SPARK_HOME:$PATH
  export PYTHONPATH=$SPARK_HOME/python:$PYTHONPATH

- Now open up a Jupyter notebook from the bare metal instance and type:
  from pyspark import SparkContext
  sc=SparkContext()
  (if it is installed correctly it should go through fine)

SQL Database in the Cloud

	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
Capacity	Petabytes +	Gigabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Persistent Hashmap	Key-value(s), HBase API	Relational
Read	Have to copy to local disk	SELECT rows	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row	put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud	Structured data from AppEngine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

What is Cloud SQL?



Rentals Case Study

- upload mtg04.zip to your Cloud Shell (not your instance)
- start your Cloud Shell and unzip mtg04
- move into the rentals-sql folder (within the mtg04 folder)
- check out the files in the cloudsql folder
- try:
  head *.csv
- move these files into a bucket, how?

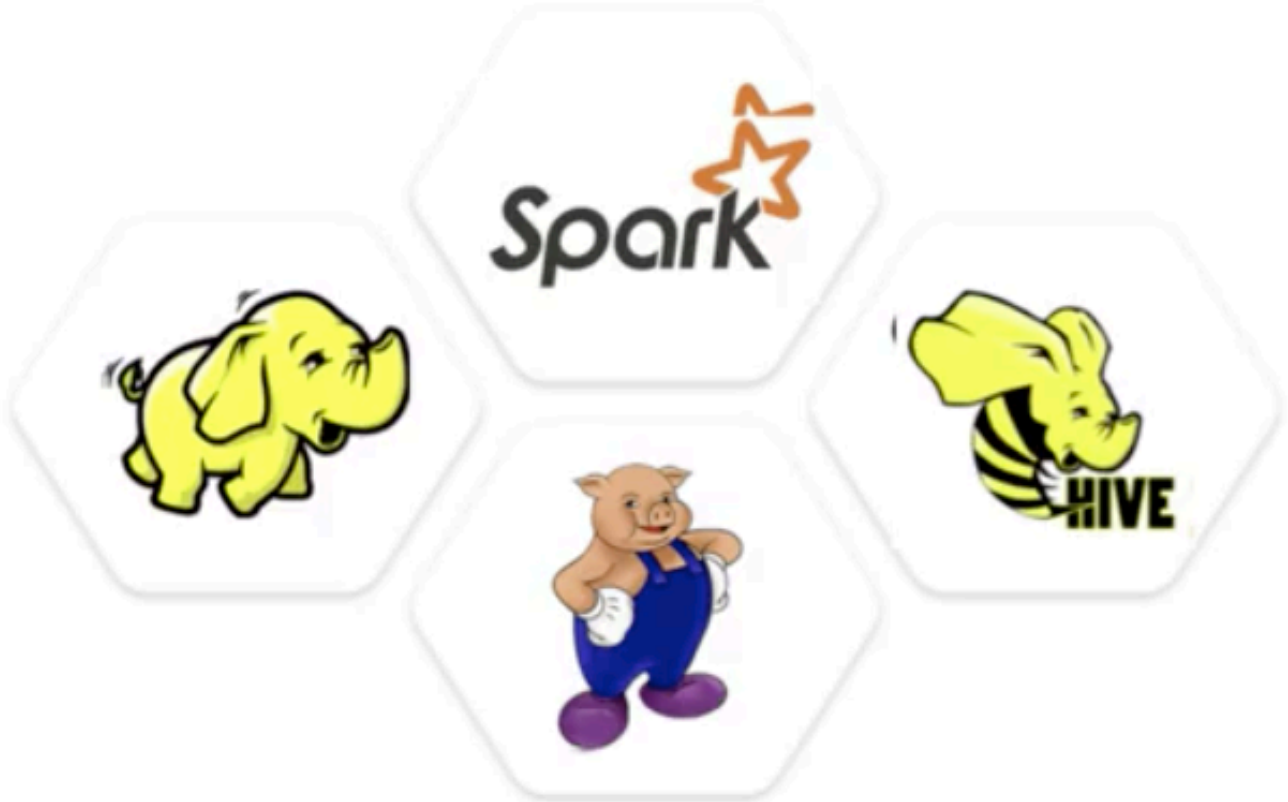
- now create a Cloud SQL instance named rentals
  (when creating instance, choose second generation, and authorize networks under show configuration options, run script to ip address) – note the password
- keep track of public IP address for SQL instance
- go into the SQL instance and click import, and select SQL file
- go into SQL instance and click import again, and use recommendation spark for the csv files

Rentals Case Study

- You can now use the mysql cli from your Cloud Shell:
  mysql --host=PUBLIC_IP –user=root --password
- Enter your password

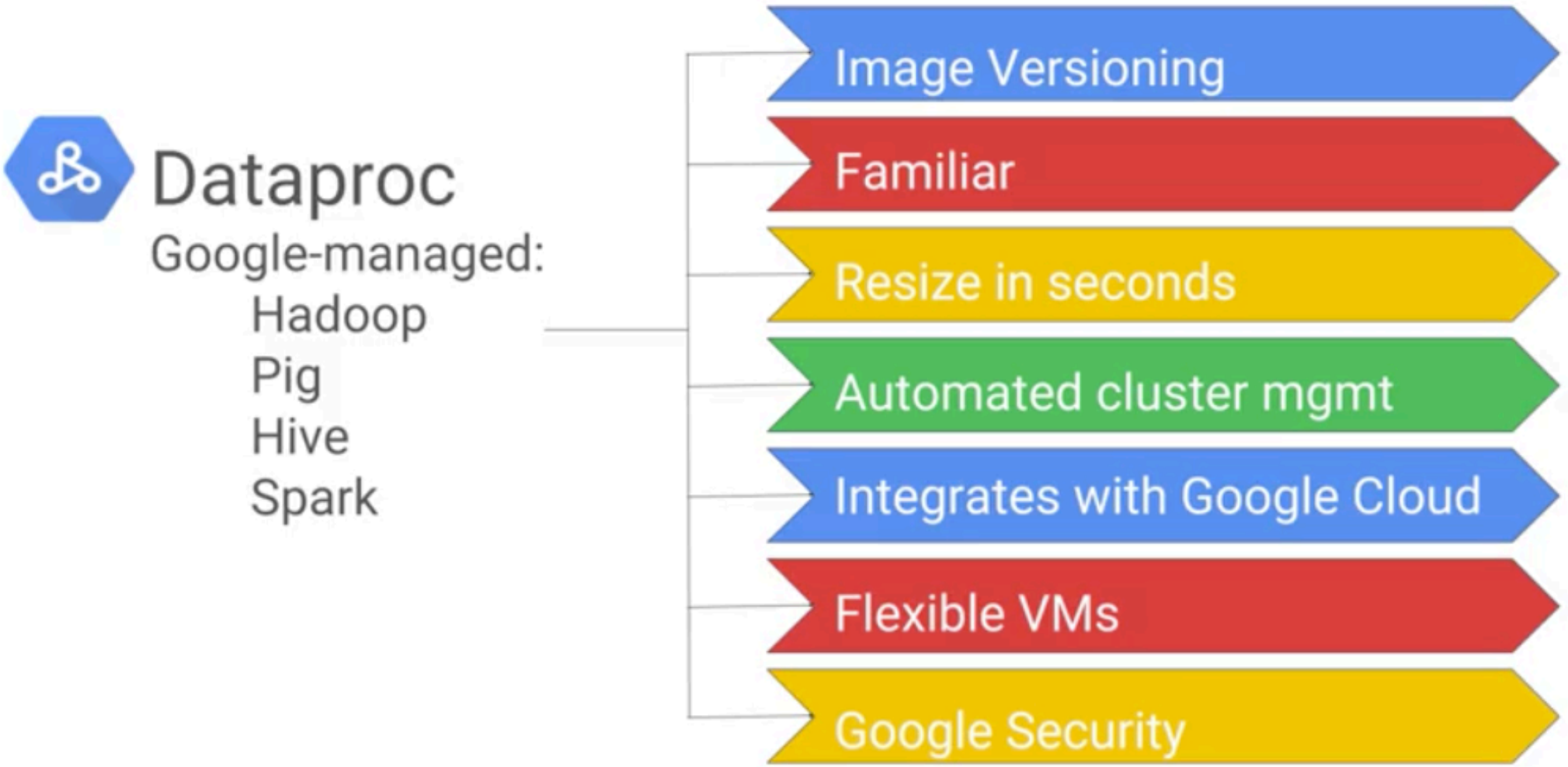
- SQL time:
  use recommendation_spark;
  show tables;
  select * from Rating;
  select * from Accomodation where type = ‘castle’ and price < 1500;

Big Data Open-Source Ecosystem



http://hadoop.apache.org/
http://pig.apache.org/
http://hive.apache.org/
http://spark.apache.org/

What is Dataproc? With Dataproc you can separate the storage and compute lifecycles.



Let's use Dataproc.

bash authorize_dataproc.sh
nano train_and_apply.py
gsutil cp train_and_apply.py gs://yourbucket
submit job

now navigate to rentals-sql
bash authorize_cloudshell.sh
mysql --host=.. --user=root –password
use recommendation_spark;

```
select r.userid, r.accoid, r.prediction, a.title, a.location, a.price,
    a.rooms, a.rating, a.type from Recommendation as r,
    Accommodation as a where r.accoid=a.id and r.userid=10;
```

BigQuery & Datalab & Ungit

Review & Practice Problems