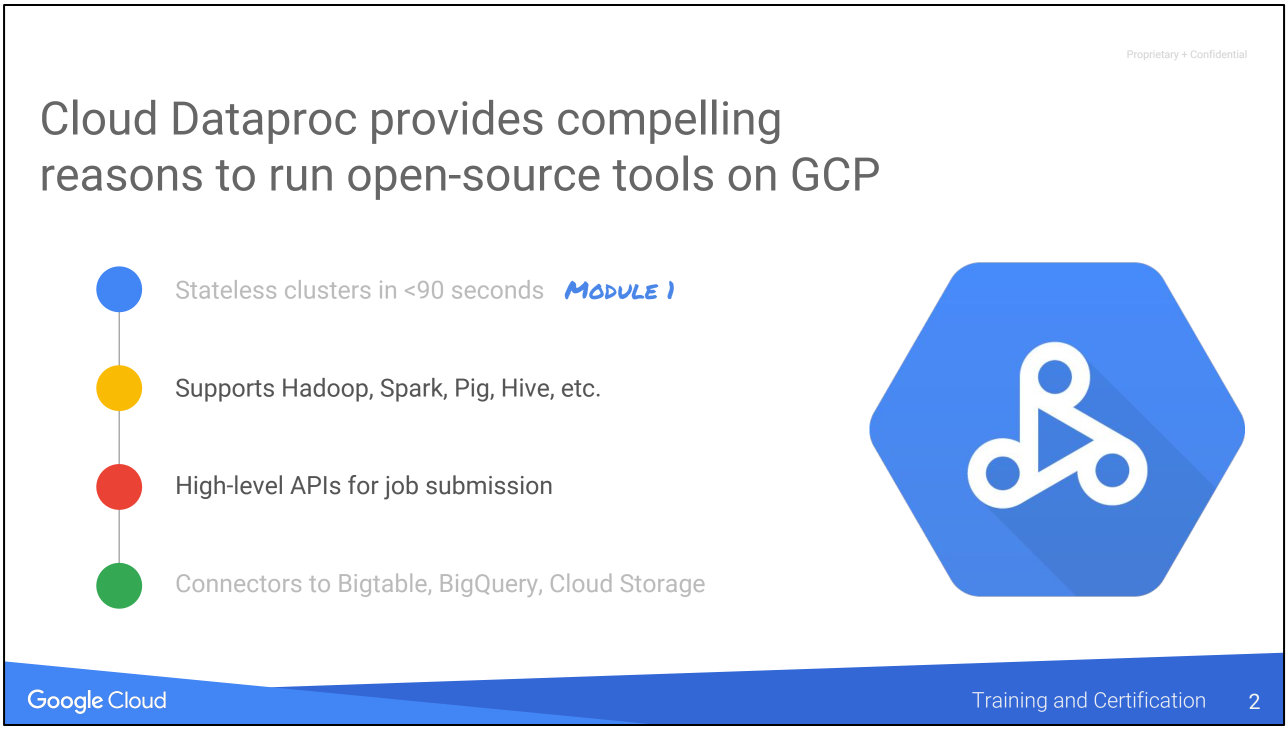
# Running Dataproc jobs

## Data Engineering on Google Cloud Platform

**Notes:**

25 slides + 1 lab: 1 hour



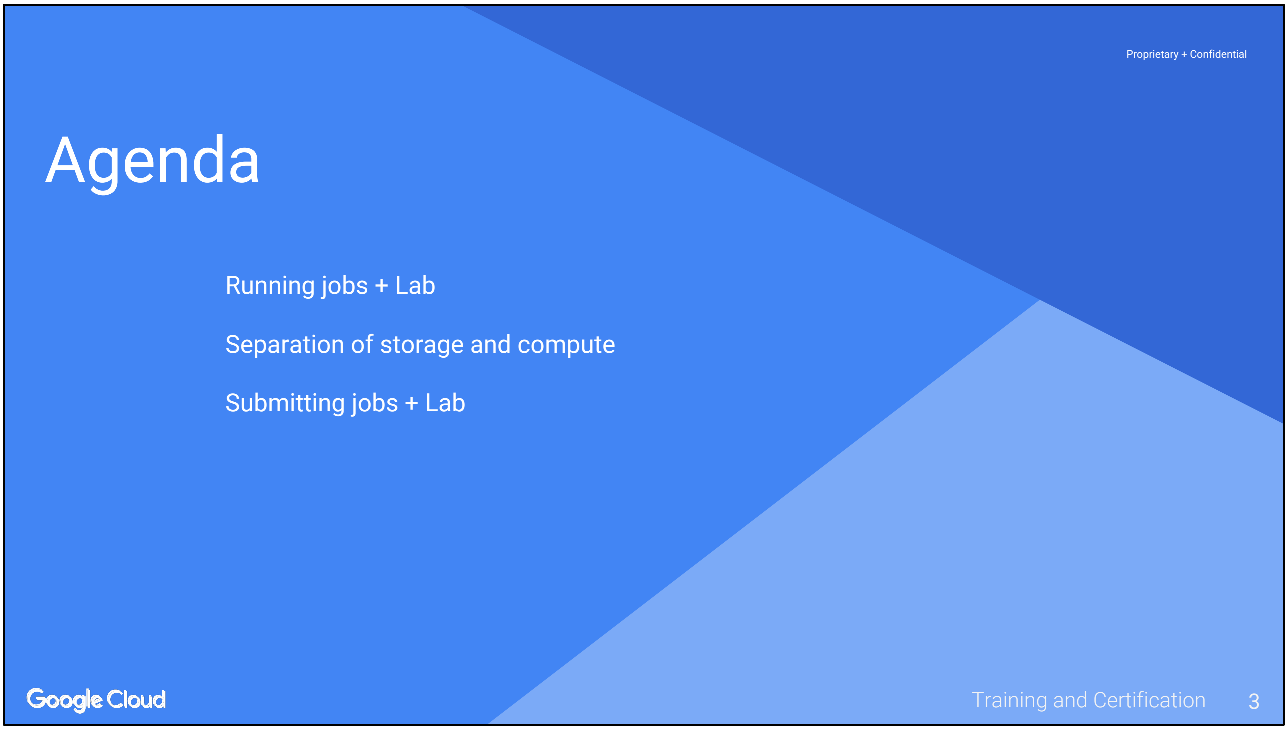
# Cloud Dataproc provides compelling reasons to run open-source tools on GCP

- Stateless clusters in <90 seconds MODULE 1
- Supports Hadoop, Spark, Pig, Hive, etc.
- High-level APIs for job submission
- Connectors to Bigtable, BigQuery, Cloud Storage

**Notes:**

We have already looked at #1.

Let’s look at #2 and #3 here. Starting with #2.



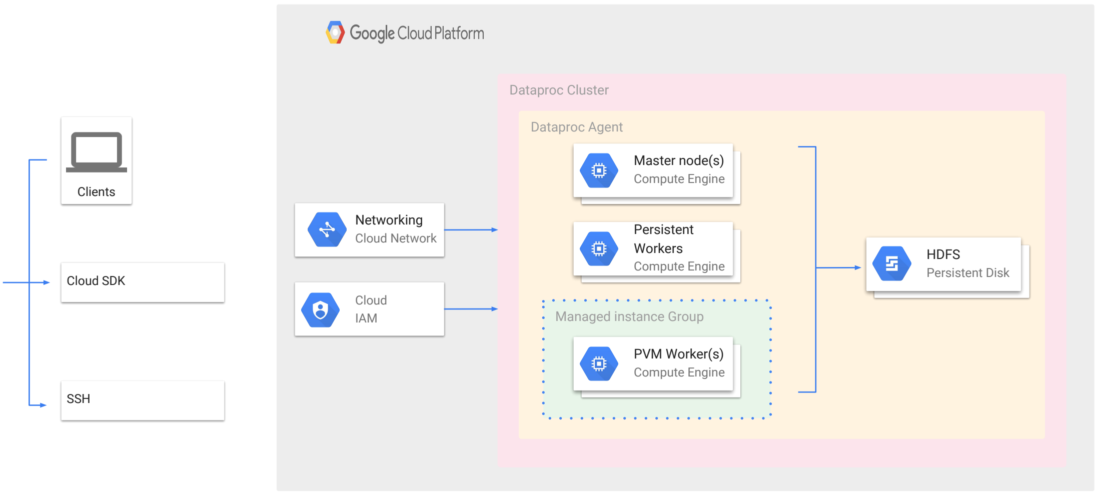
# Agenda

Running jobs + Lab

Separation of storage and compute

Submitting jobs + Lab

# Can SSH to cluster and run Pig/Spark

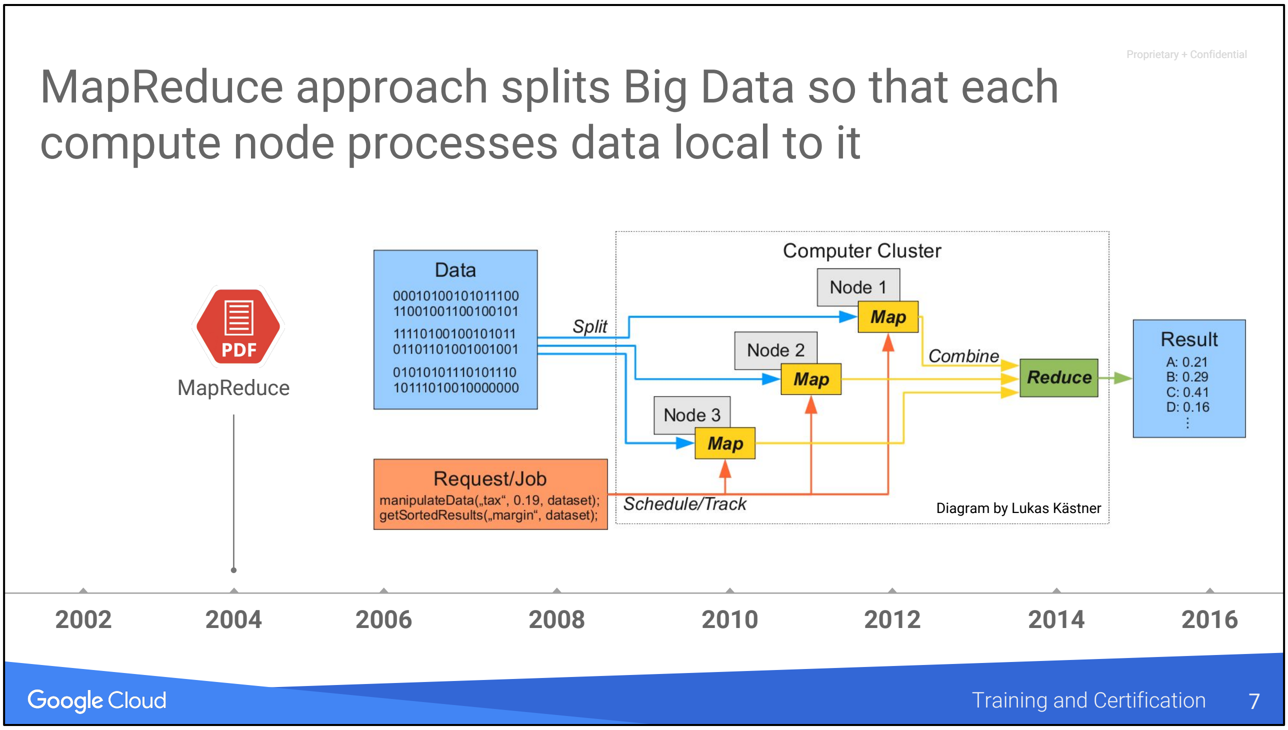


# Lab - Leveraging Unstructured Data : Part 2

- SSH into the cluster to run Pig and Spark jobs interactively
- Work with HDFS

# Agenda

Separation of storage and compute



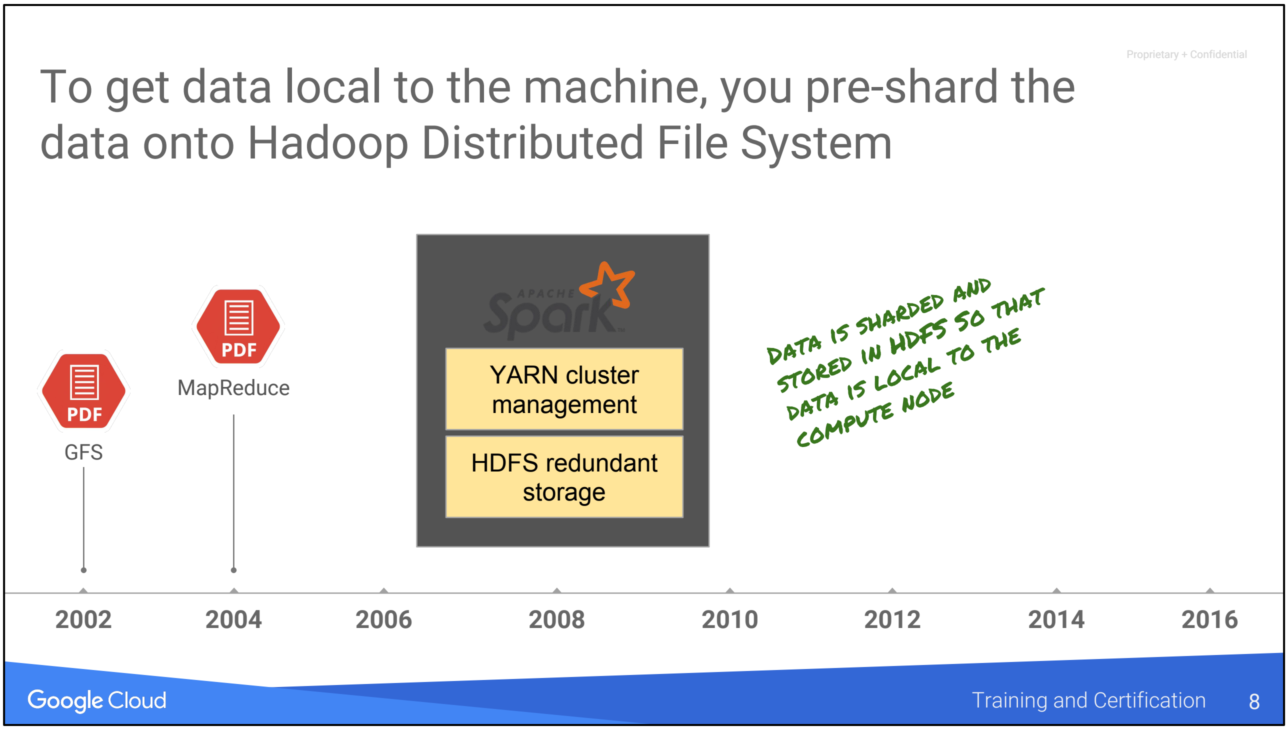
# MapReduce approach splits Big Data so that each compute node processes data local to it

Diagram by Lukas Kästner

**Notes:**

This slide was also in Chapter 1, so just mention that they’ve already seen this.

Diagram source: https://www.flickr.com/photos/lkaestner/4861146813
cc-by-sa Lukas Kastner



# To get data local to the machine, you pre-shard the data onto Hadoop Distributed File System

GFS

MapReduce

APACHE Spark

YARN cluster management

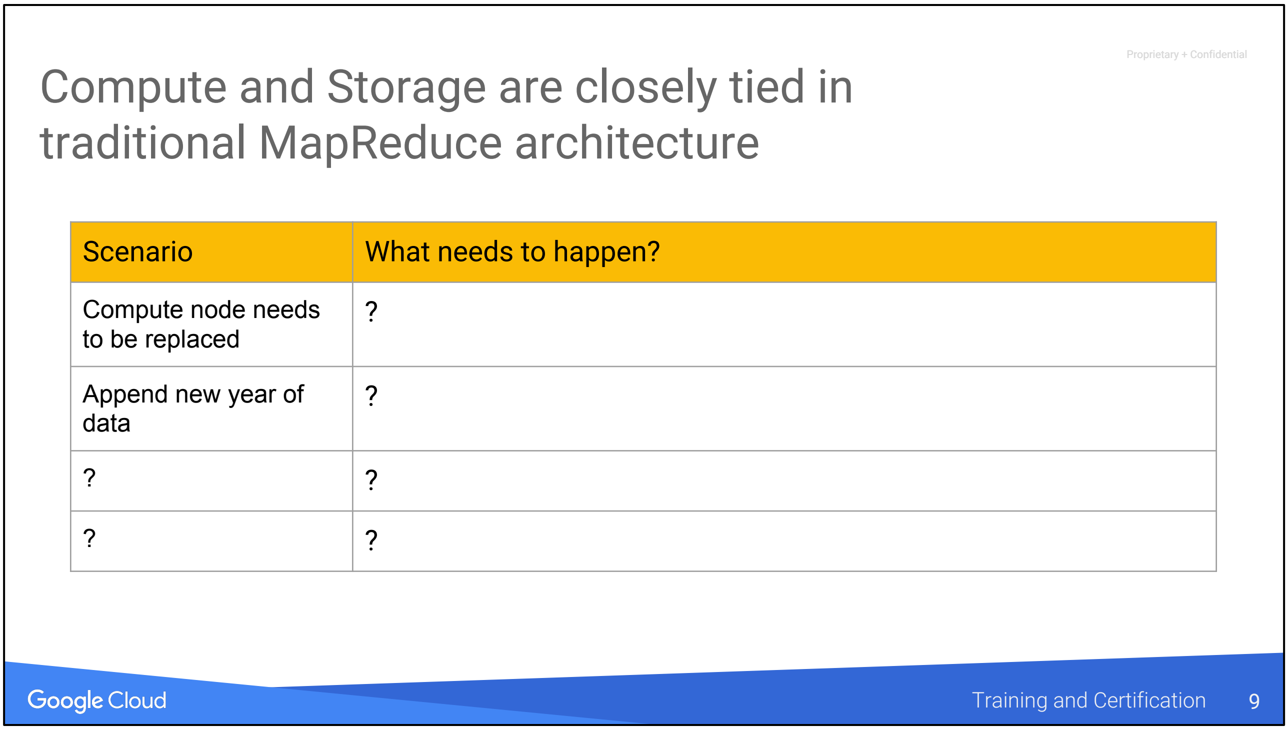
HDFS redundant storage

DATA IS SHARDED AND STORED IN HDFS SO THAT DATA IS LOCAL TO THE COMPUTE NODE

2002
2004
2006
2008
2010
2012
2014
2016

**Notes:**

HDFS is based on the 2002 paper from Google on Google File System.



# Compute and Storage are closely tied in traditional MapReduce architecture

| Scenario | What needs to happen? |
|---|---|
| Compute node needs to be replaced | ? |
| Append new year of data | ? |
| ? | ? |
| ? | ? |

**Notes:**

Ask the class what problems this leads to. Have them think about a scenario.

Example scenario: You split your data into 10 nodes. One node goes down. You bring in a new replacement. What has to happen? At least some part of the data needs to be copied onto the new nodes before jobs have to be partitioned.

Example scenario: The data changes (maybe you need to change formats or append a new year of data). What needs to happen?

Proprietary + Confidential

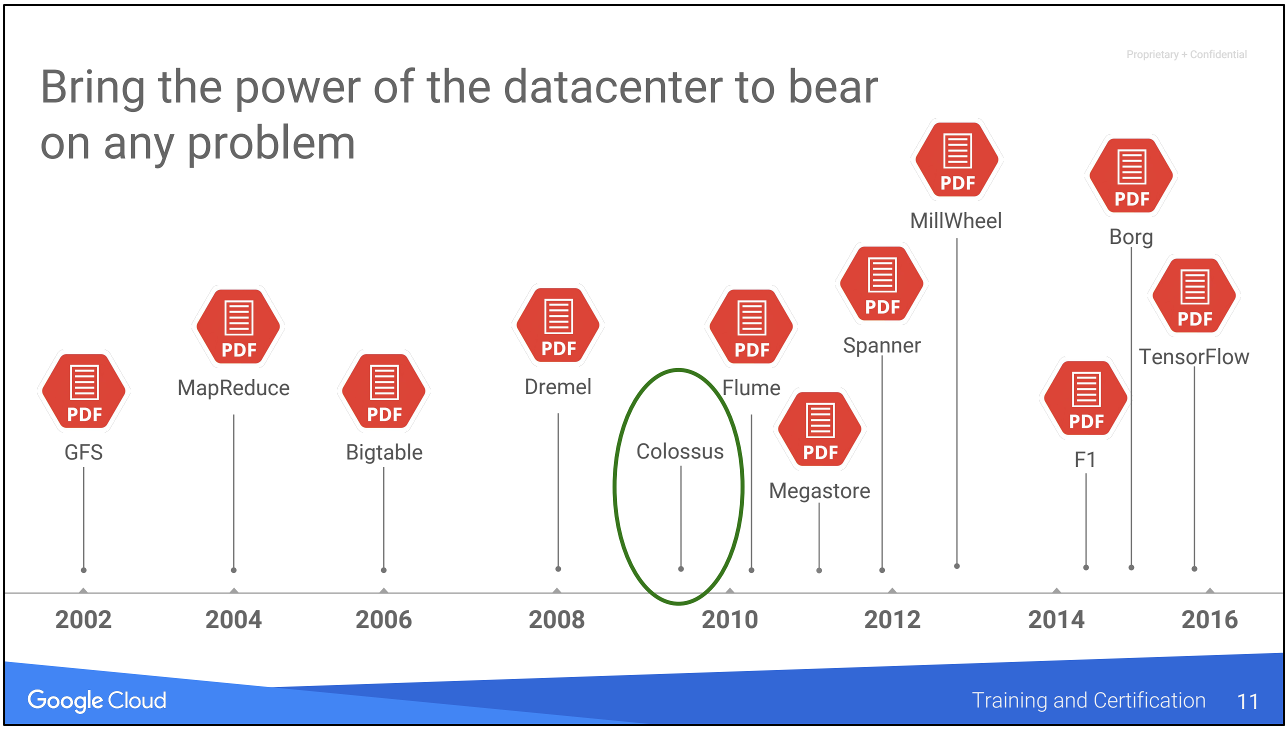
# Need to provision increased resources quickly

**Notes:**

Beyond the obvious question of deciding how much to provision, the larger issue is that of the risk/cost of experimentation.

If they knew for sure what they have to do, how to do it, and what resources they need for it, enterprises would be able to provision large amounts of resources. But the reality is that you have to experiment. It’s not about how much you can invest, it’s how quickly you can iterate.

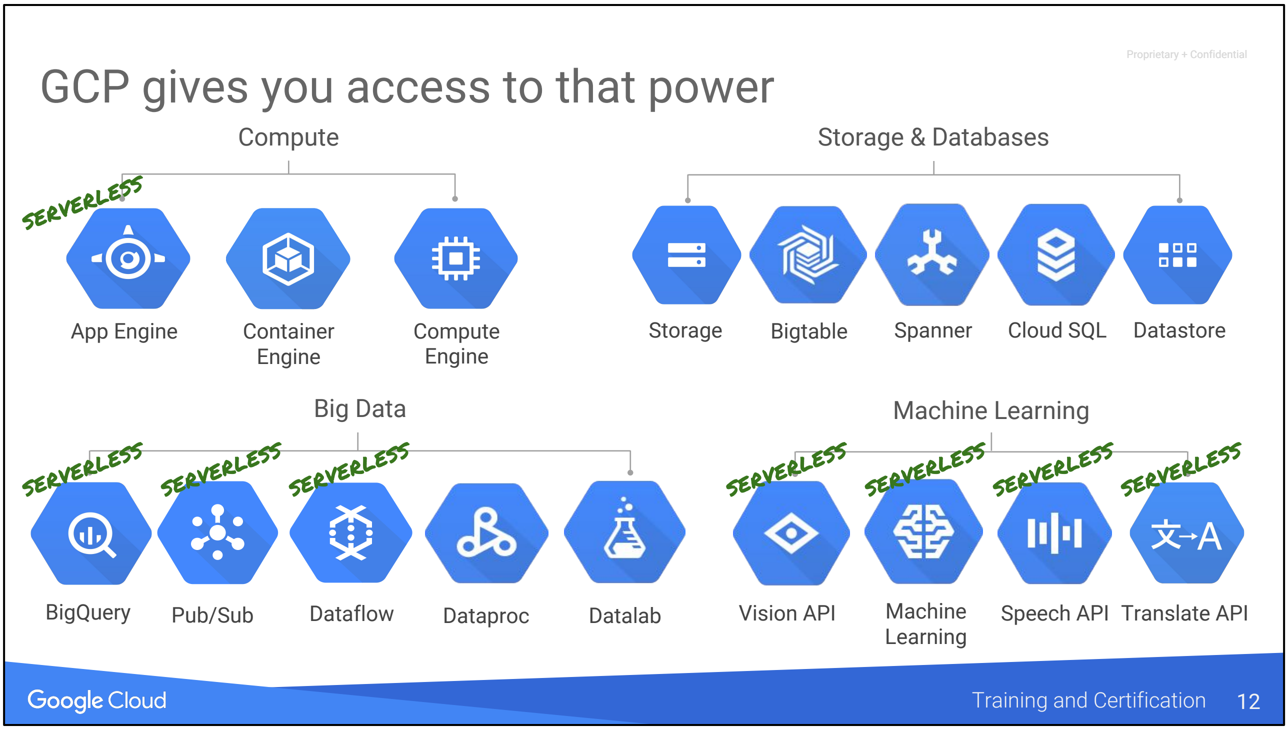
Image source: I (vbp@) took it myself.



# Bring the power of the datacenter to bear on any problem

**Notes:**

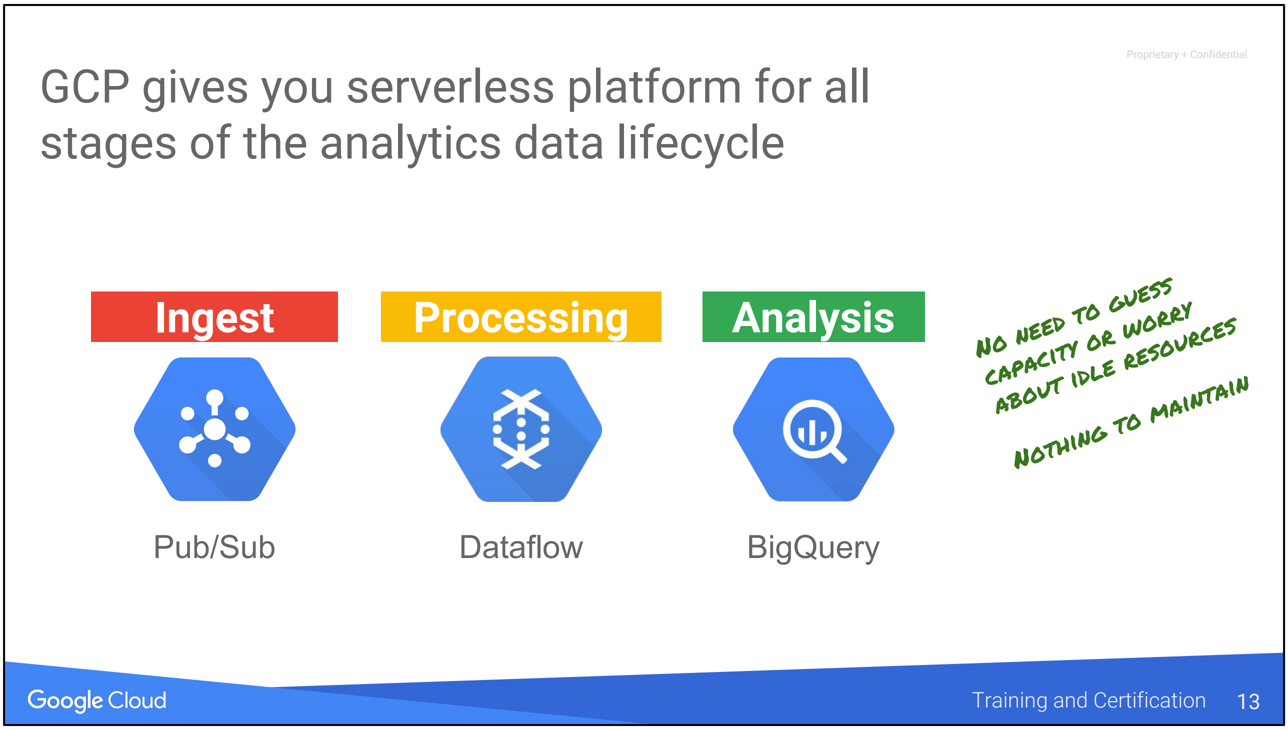
Colossus, the replacement for GFS, is the key innovation and led to a bunch of serverless offerings. It’s in our datacenter and enables you to not have to shard data. Instead, you have a global filesystem that offers petabit/second bisection bandwidth. Yes, Colossus has not been published … public information on it is scarce and consists of an unofficial copy of a slide deck by Andrew Fikes: https://www.systutorials.com/3306/storage-architecture-and-challenges/



# GCP gives you access to that power

**Notes:**

The Big Data and ML offerings are serverless (except for Dataproc & Datalab because they are based on OSS -- Hadoop ecosystem and Jupyter respectively). The APIs are of course serverless although you tend not to think of them as serverless offerings.



# GCP gives you serverless platform for all stages of the analytics data lifecycle

Ingest

Pub/Sub

Processing

Dataflow

Analysis

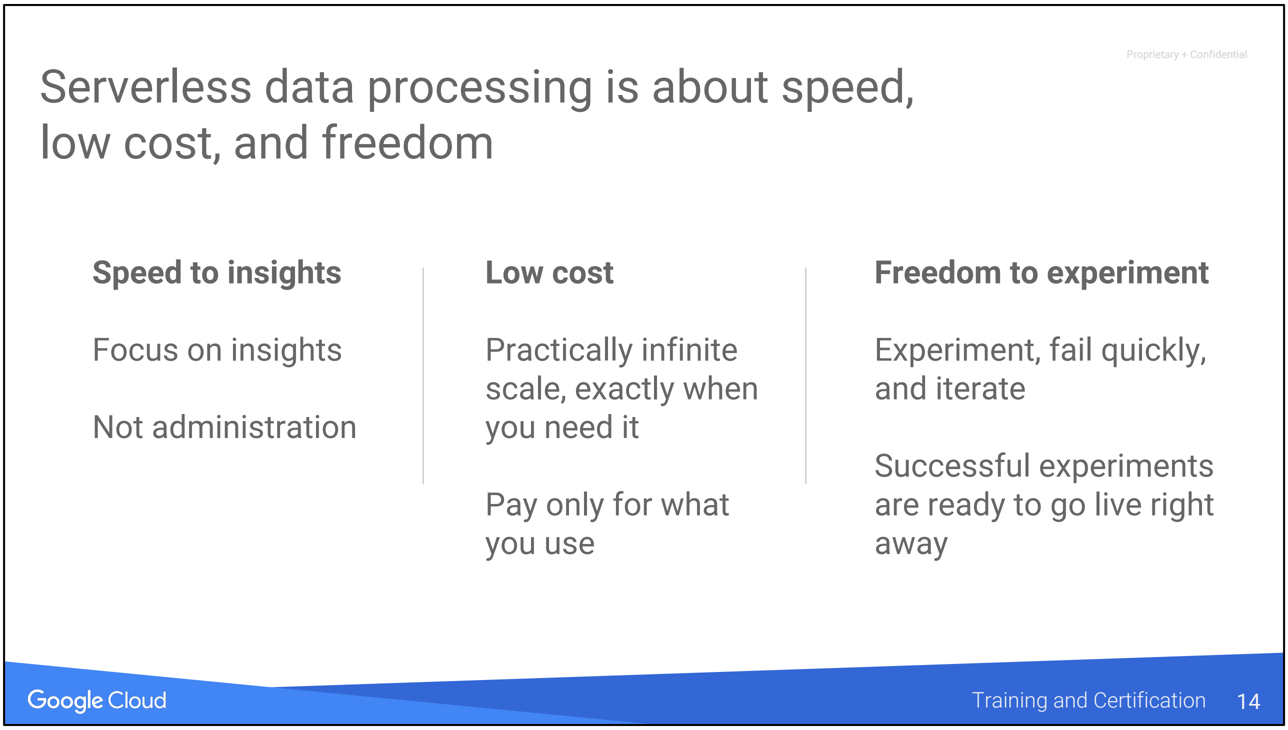
BigQuery

NO NEED TO GUESS CAPACITY OR WORRY ABOUT IDLE RESOURCES

NOTHING TO MAINTAIN

**Notes:**

In particular, the three Big Data serverless offerings are ...



# Serverless data processing is about speed, low cost, and freedom

**Speed to insights**

Focus on insights

Not administration

**Low cost**

Practically infinite scale, exactly when you need it

Pay only for what you use

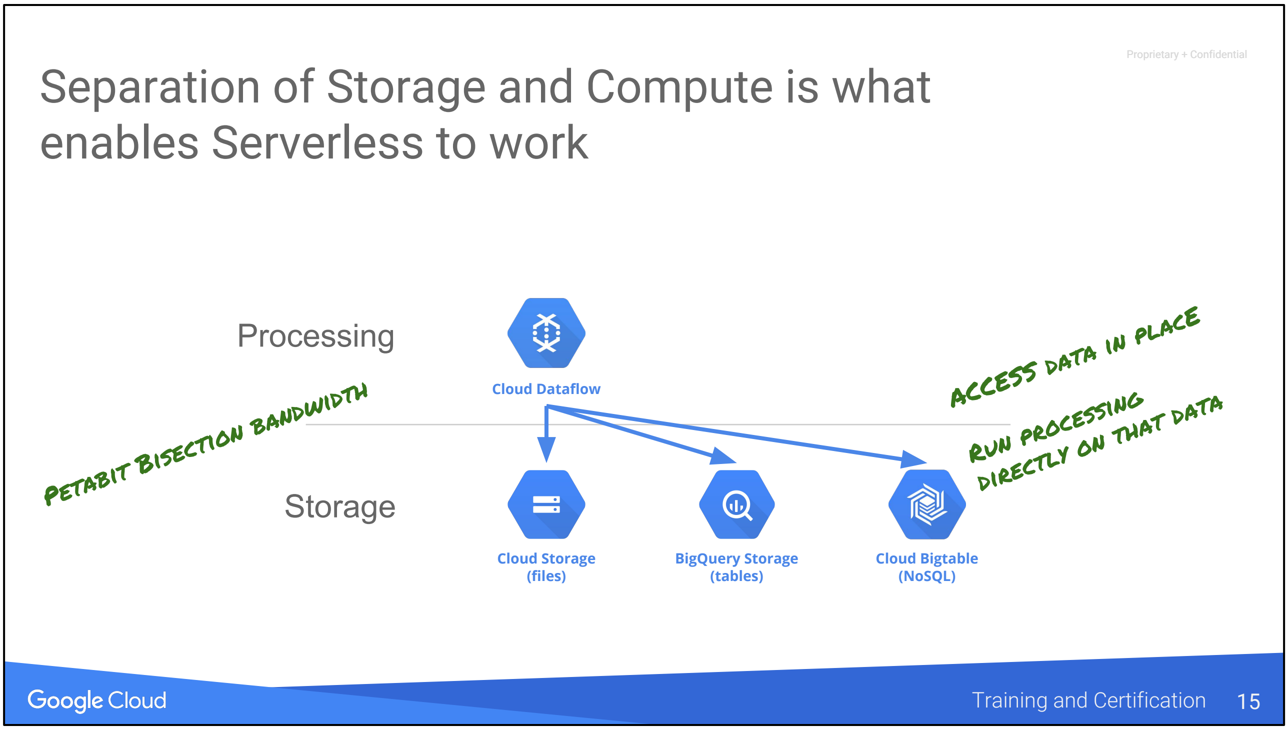
**Freedom to experiment**

Experiment, fail quickly, and iterate

Successful experiments are ready to go live right away

**Notes:**

Summary slide for this section: Not just about low-cost, but also about speed and freedom.



# Separation of Storage and Compute is what enables Serverless to work

Processing

Cloud Dataflow

Storage

Cloud Storage (files)

BigQuery Storage (tables)

Cloud Bigtable (NoSQL)

PETABIT BISECTION BANDWIDTH

ACCESS DATA IN PLACE

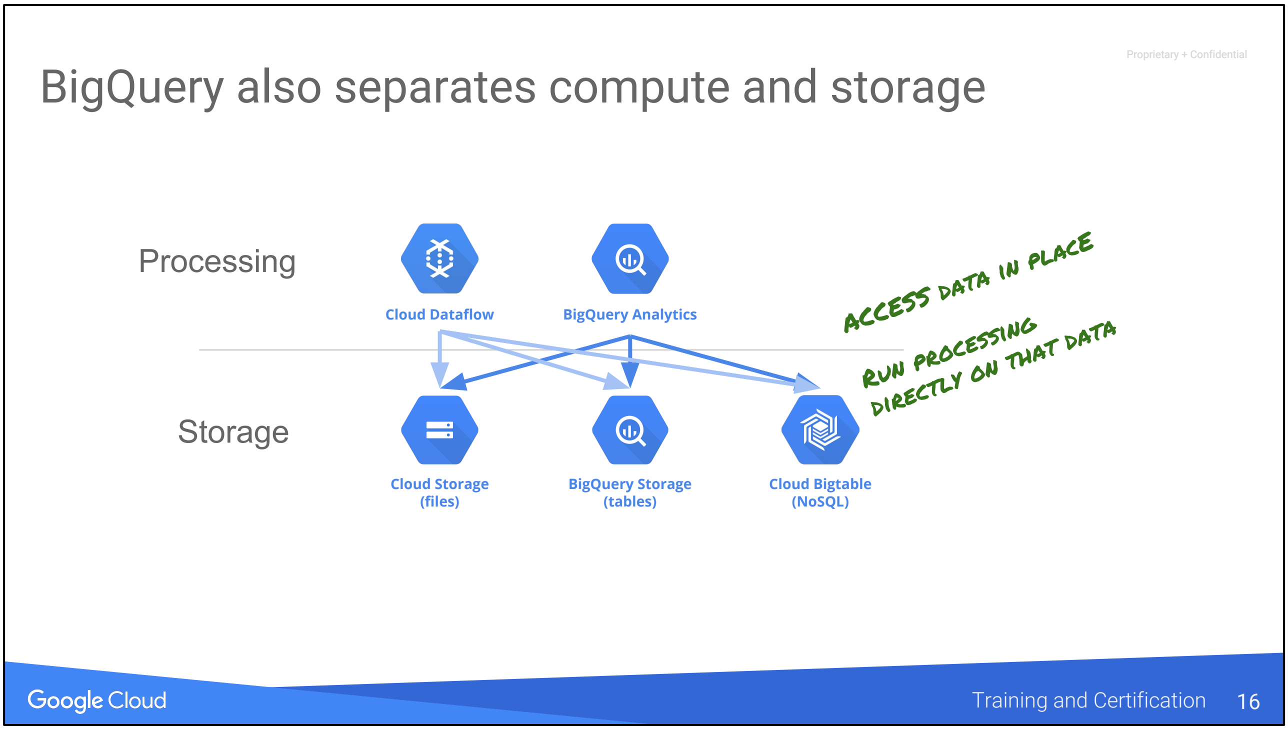
RUN PROCESSING DIRECTLY ON THAT DATA

**Notes:**

Separation of storage and compute is what enables “serverless to work” -- Dataflow can read use any of these as source/sink. This sort of direct read is efficient because of very high sustained read speed from Cloud Storage -- any two computers in data center are connected by very fast network.

Keep as much data as you want, economically.

Share data in place, no more FTP and copying.



Proprietary + Confidential

## BigQuery also separates compute and storage

Processing

Cloud Dataflow

BigQuery Analytics

Storage

Cloud Storage
(files)

BigQuery Storage
(tables)

Cloud Bigtable
(NoSQL)

ACCESS DATA IN PLACE

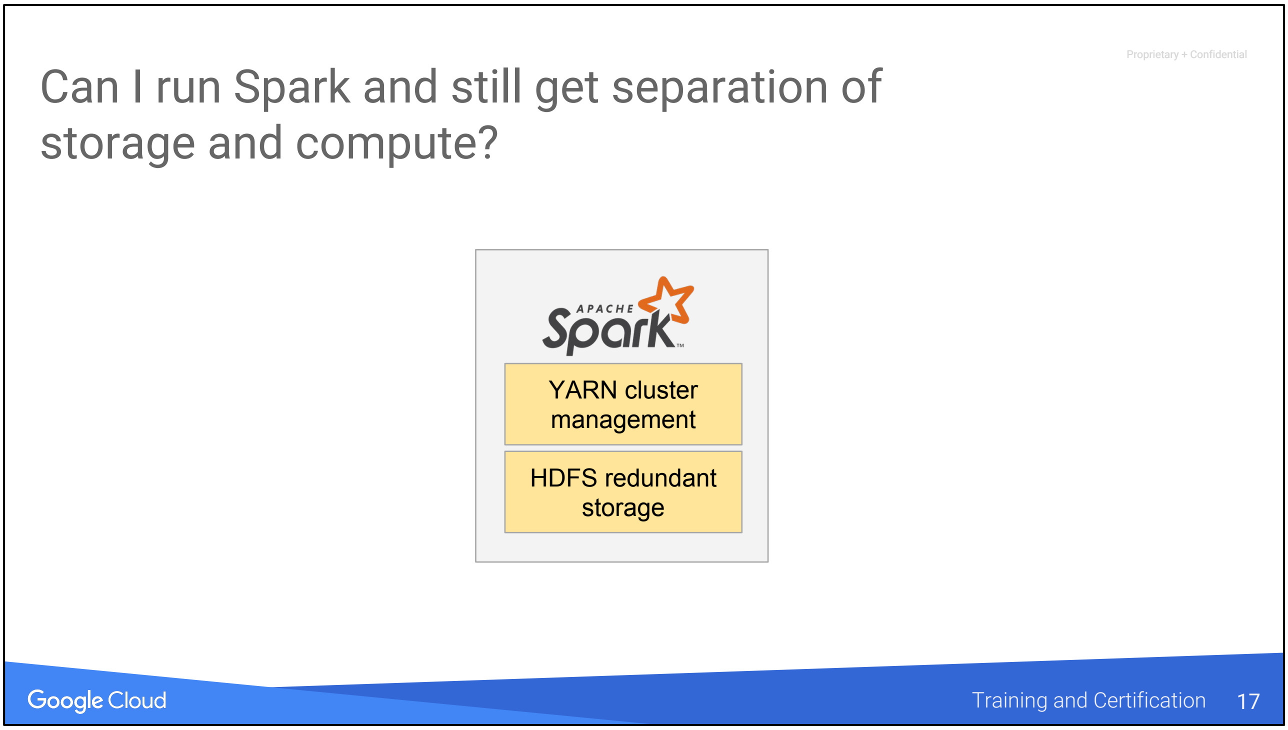
RUN PROCESSING
DIRECTLY ON THAT DATA

**Notes:**

Access any storage system from any processing tool.

Compare and contrast bq’s data separation w/ typical DB storage mgmt system.

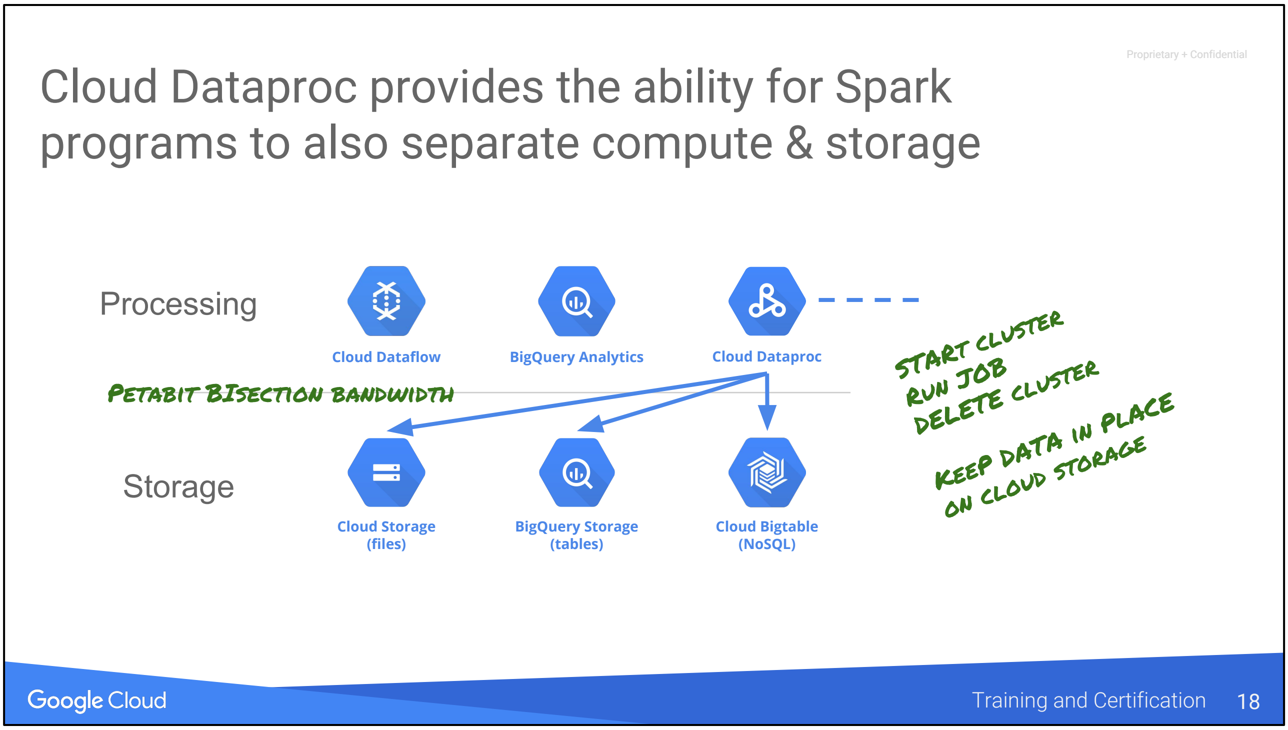
BigQuery is “just” a query engine. It can query csv files on cloud storage also, for example.



# Can I run Spark and still get separation of storage and compute?

**Notes:**

But what if I want to run Spark programs? How can I get separation of compute & storage. Spark runs on Hadoop … and Hadoop is a cluster-aware piece of software … we need to take our data and split it into pieces and store them on cluster so that data is local to compute … but then we are limited by the number of processing nodes or the number of storage nodes. These are not independent



# Cloud Dataproc provides the ability for Spark programs to also separate compute & storage

Processing

Cloud Dataflow

BigQuery Analytics

Cloud Dataproc

PETABIT BISECTION BANDWIDTH

Storage

Cloud Storage (files)

BigQuery Storage (tables)

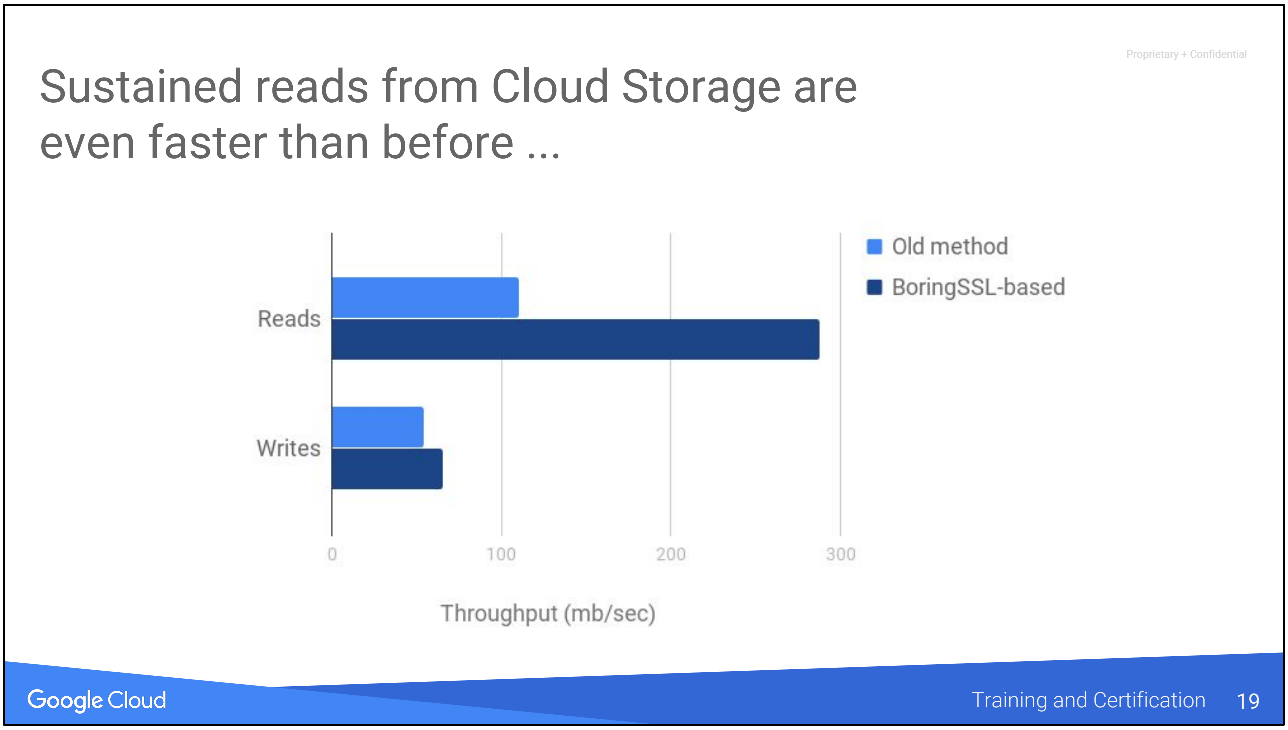
Cloud Bigtable (NoSQL)

START CLUSTER
RUN JOB
DELETE CLUSTER

KEEP DATA IN PLACE
ON CLOUD STORAGE

**Notes:**

Just change all your input urls from hdfs:// to gs:// …
The reason you can do this is the speed of the inter-networking ….
Cloud Dataproc is Spark/Hadoop “the Cloud way”
Deploy cluster in ~90 seconds
Pay by the minute



# Sustained reads from Cloud Storage are even faster than before ...

**Notes:**

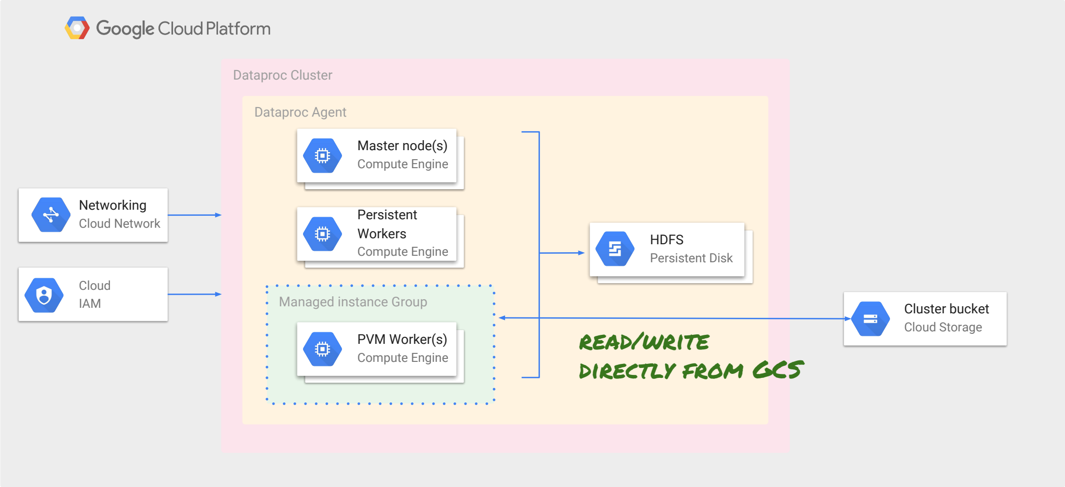
The impact of the PB/s networking and software improvements is that sustained reads are very fast. You can keep things in GCS.

Old: 2016
New: 2017

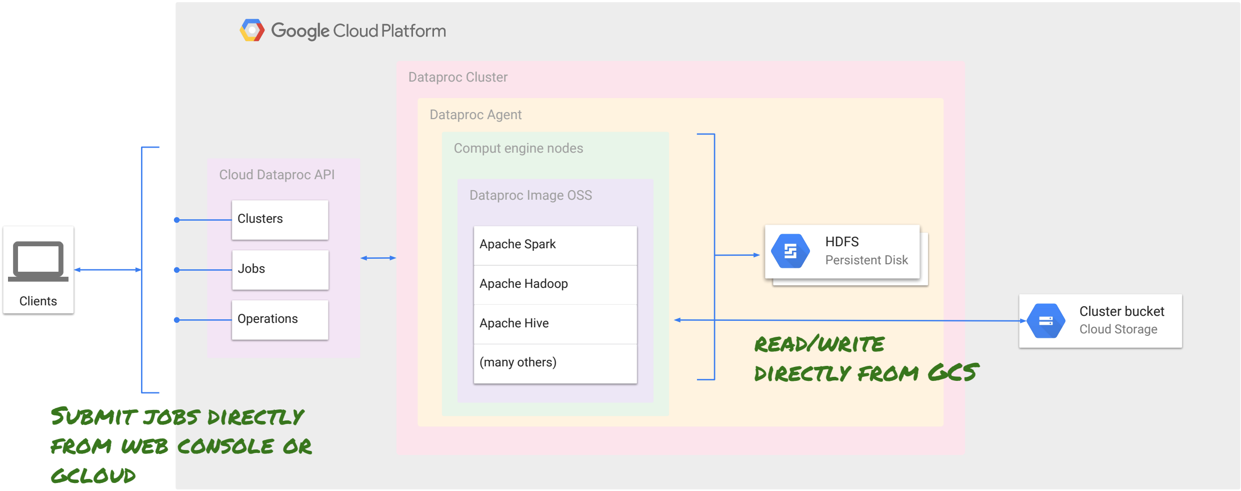
# Agenda

Submitting jobs + Lab

# Cloud Dataproc hardware architecture



# Cloud Dataproc software architecture



Google Cloud Platform

Dataproc Cluster

Dataproc Agent

Comput engine nodes

Dataproc Image OSS

Apache Spark

Apache Hadoop

Apache Hive

(many others)

Cloud Dataproc API

Clusters

Jobs

Operations

Clients

HDFS
Persistent Disk

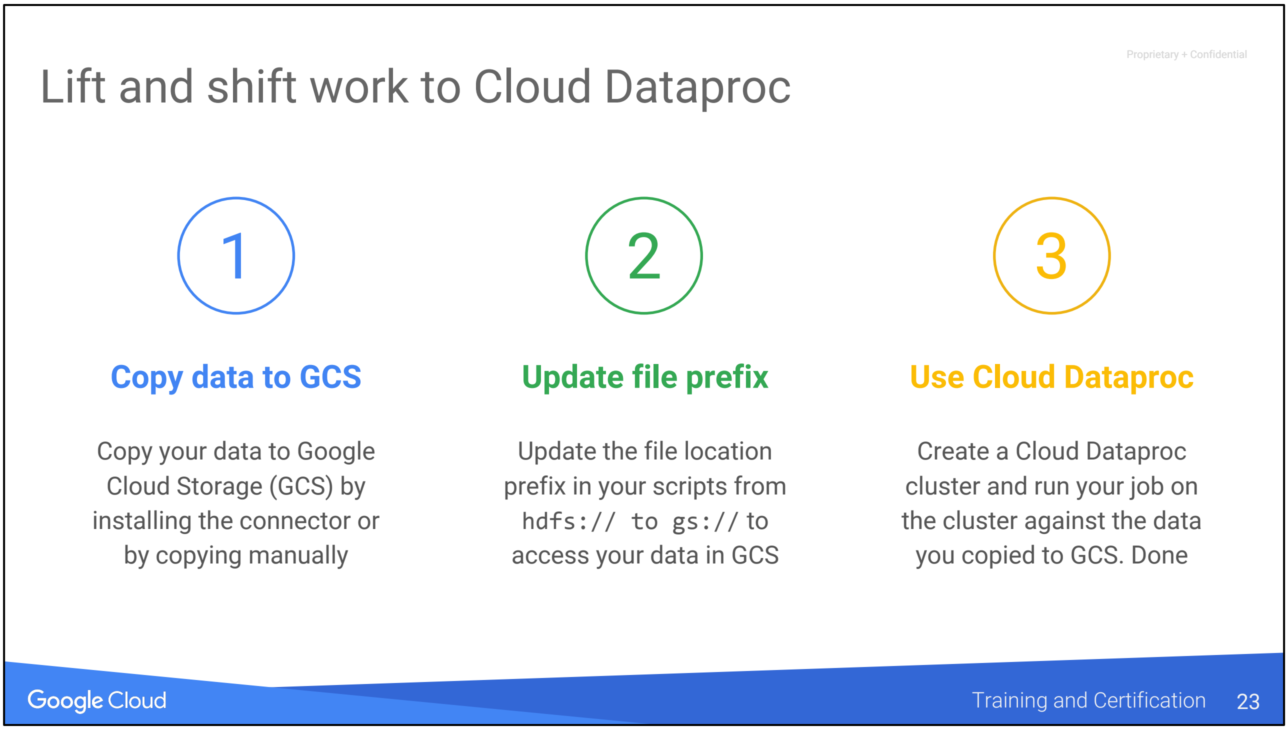
Cluster bucket
Cloud Storage

READ/WRITE
DIRECTLY FROM GCS

SUBMIT JOBS DIRECTLY
FROM WEB CONSOLE OR
GCLOUD

**Notes:**

Can directly read/write to GCS from Spark, Pig, etc.
Submit jobs



# Lift and shift work to Cloud Dataproc

1

**Copy data to GCS**

Copy your data to Google Cloud Storage (GCS) by installing the connector or by copying manually

2

**Update file prefix**

Update the file location prefix in your scripts from `hdfs:// to gs://` to access your data in GCS

3

**Use Cloud Dataproc**

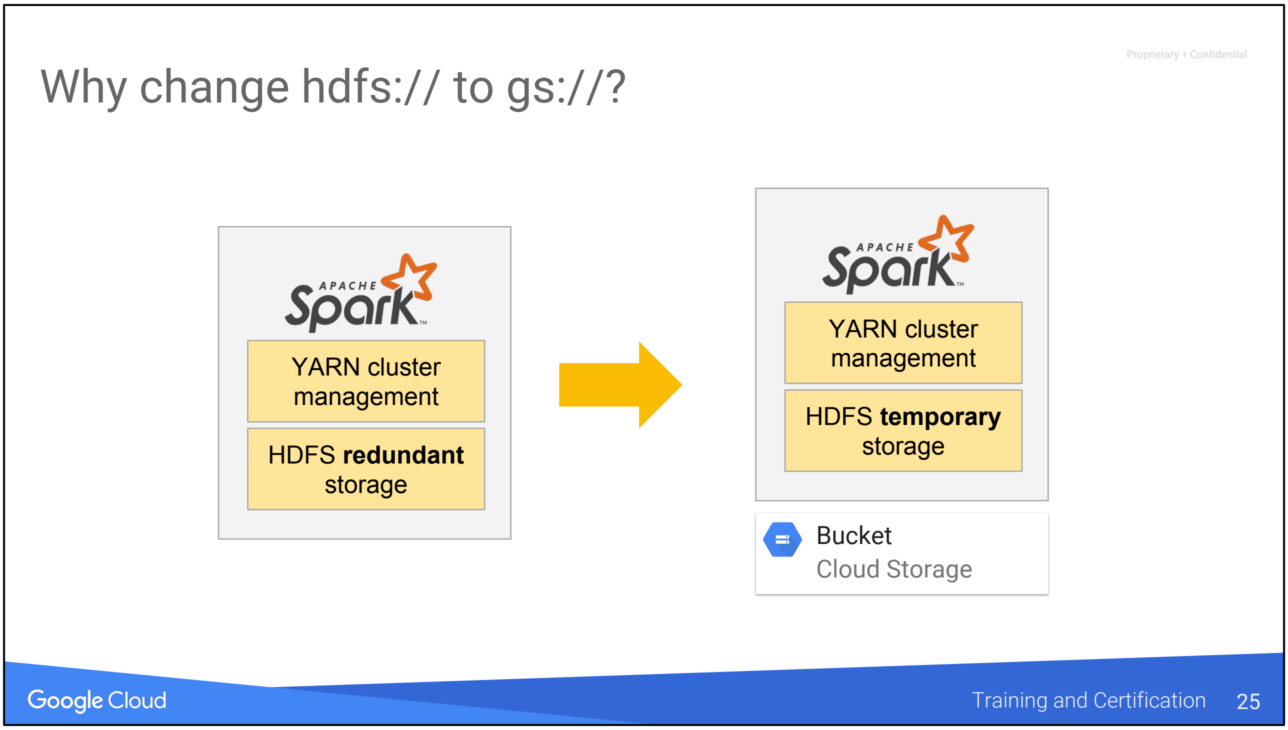
Create a Cloud Dataproc cluster and run your job on the cluster against the data you copied to GCS. Done

# Migrating code

- In most cases, you only need to update jobs so they read from Google Cloud Storage (`gs://`) instead of HDFS

```
textFile = sc.textFile("~~hdfs~~ gs://...") # Read data

# Creates a DataFrame having a single column
df = textFile.map(lambda r: Row(r)).toDF(["line"])
errors = df.filter(col("line").like("%ERROR%"))
# Counts all the errors
errors.count()
# Counts errors mentioning MySQL
errors.filter(col("line").like("%MySQL%")).count()
# Fetches the MySQL errors as an array of strings
errors.filter(col("line").like("%MySQL%")).collect()
```



# Why change hdfs:// to gs://?

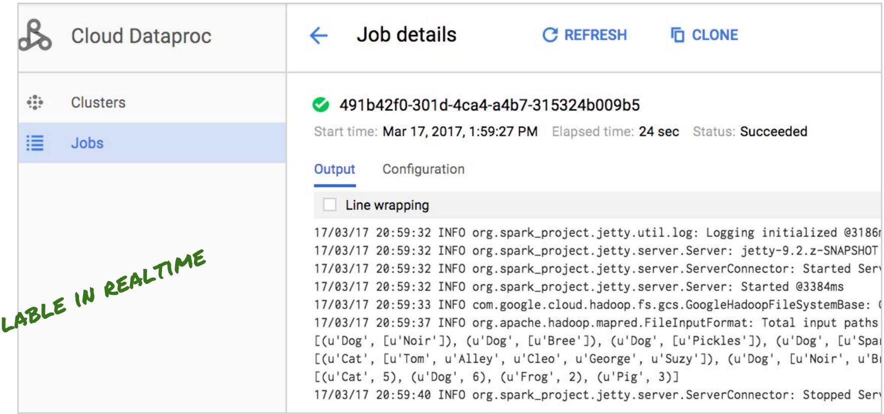
**Notes:**

Because the cluster is temporary …. We want to be able to delete the cluster when we are done.

If the first case, HDFS is the durable storage for data. We can’t delete the cluster.

In the second case, HDFS is only temporary. We can delete the cluster.

# Monitor logs of submitted jobs from web console



491b42f0-301d-4ca4-a4b7-315324b009b5

Start time: Mar 17, 2017, 1:59:27 PM Elapsed time: 24 sec Status: Succeeded

□ Line wrapping

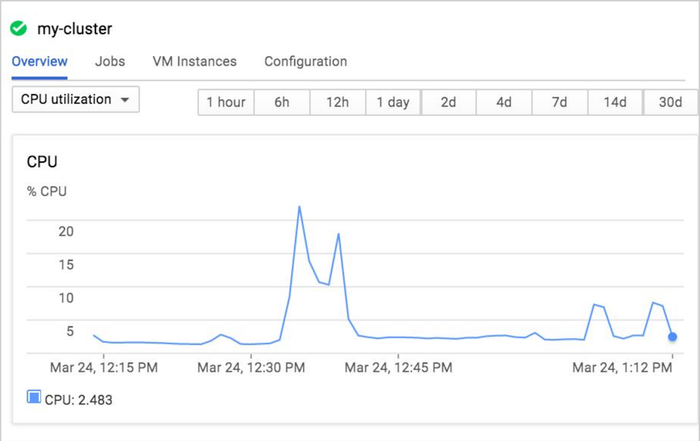
```
17/03/17 20:59:32 INFO org.spark_project.jetty.util.log: Logging initialized @3186
17/03/17 20:59:32 INFO org.spark_project.jetty.server.Server: jetty-9.2.z-SNAPSHOT
17/03/17 20:59:32 INFO org.spark_project.jetty.server.ServerConnector: Started Ser
17/03/17 20:59:32 INFO org.spark_project.jetty.server.Server: Started @3384ms
17/03/17 20:59:33 INFO com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystemBase: 
17/03/17 20:59:37 INFO org.apache.hadoop.mapred.FileInputFormat: Total input paths
[(u'Dog', [u'Noir']), (u'Dog', [u'Bree']), (u'Dog', [u'Pickles']), (u'Dog', [u'Spa
[(u'Cat', [u'Tom', u'Alley', u'Cleo', u'George', u'Suzy']), (u'Dog', [u'Noir', u'B
[(u'Cat', 5), (u'Dog', 6), (u'Frog', 2), (u'Pig', 3)]
17/03/17 20:59:40 INFO org.spark_project.jetty.server.ServerConnector: Stopped Ser
```

AVAILABLE IN REALTIME

**Notes:**

These logs are available in real-time.

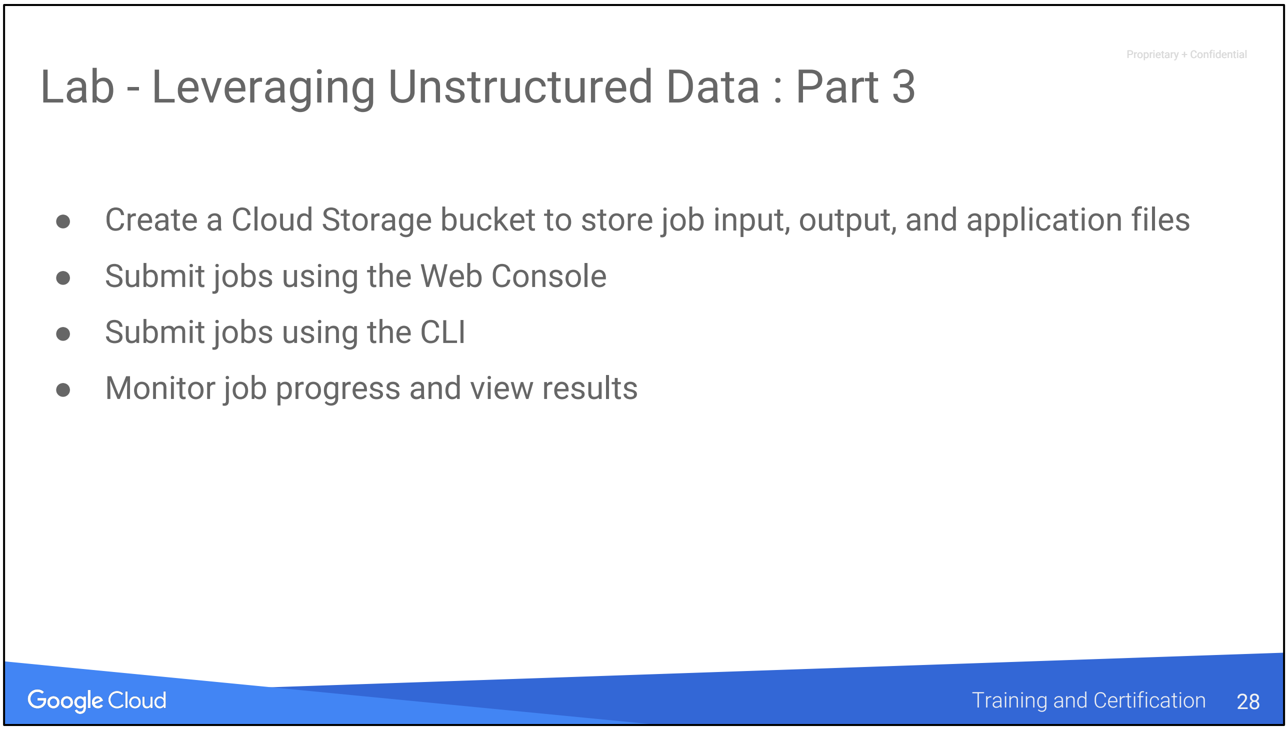
# Monitor cluster usage graphs on Web UI, Stackdriver



**Notes:**

Overall cluster usage from Dataproc page.

Individual VMs from Compute Engine VM.



# Lab - Leveraging Unstructured Data : Part 3

- Create a Cloud Storage bucket to store job input, output, and application files
- Submit jobs using the Web Console
- Submit jobs using the CLI
- Monitor job progress and view results

cloud.google.com

Images by Connie Zhou