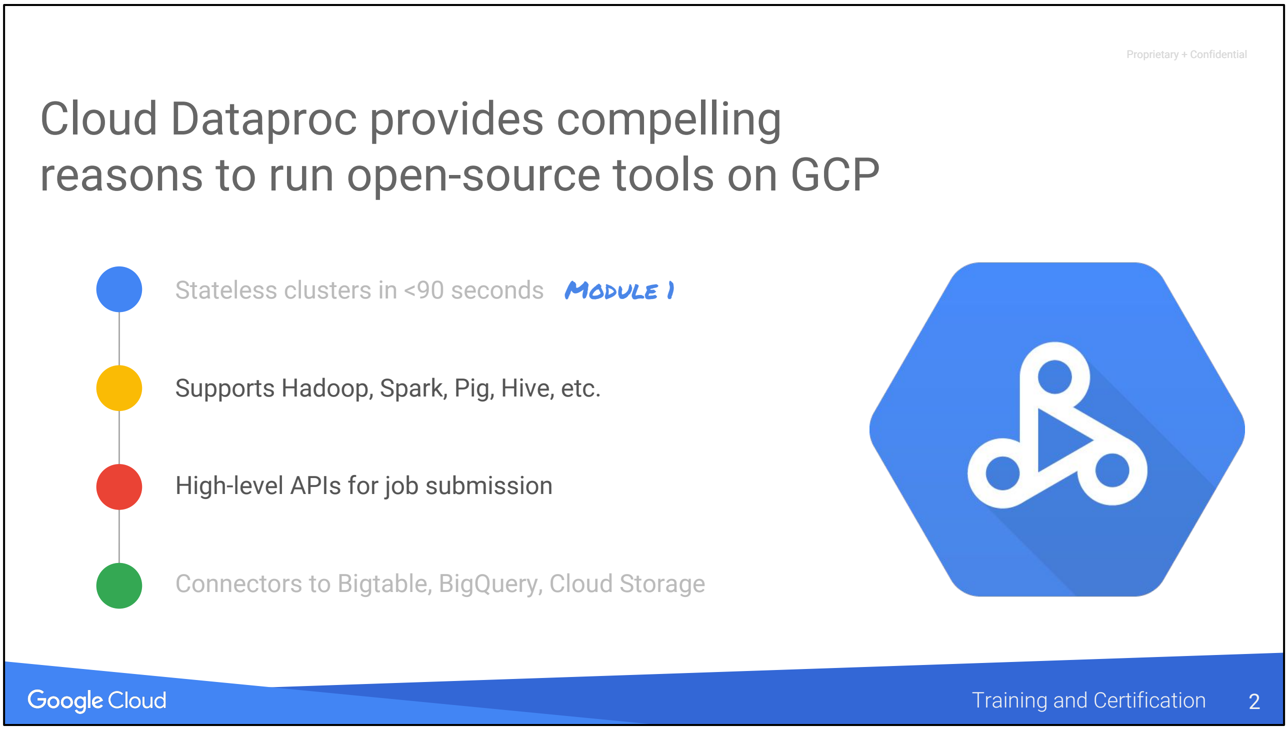
Running Dataproc jobs

Data Engineering on Google Cloud Platform

Notes:

25 slides + 1 lab: 1 hour



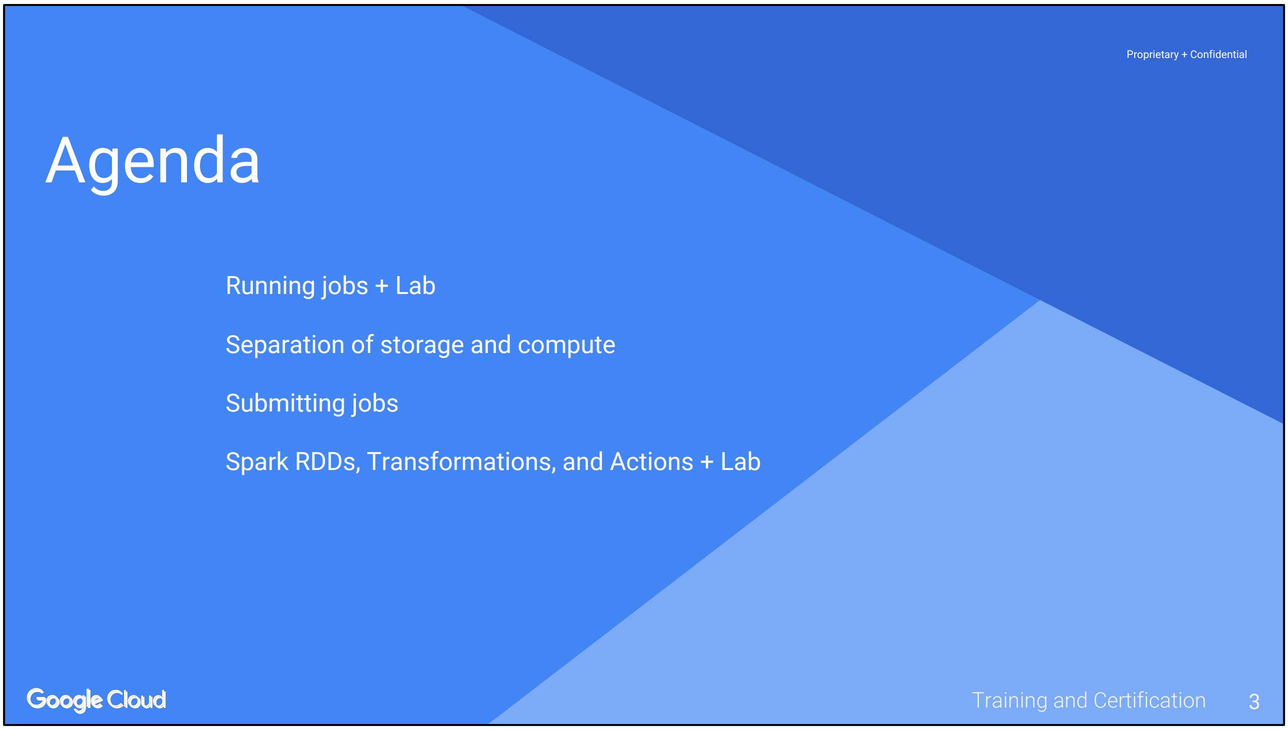
Cloud Dataproc provides compelling reasons to run open-source tools on GCP

- 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

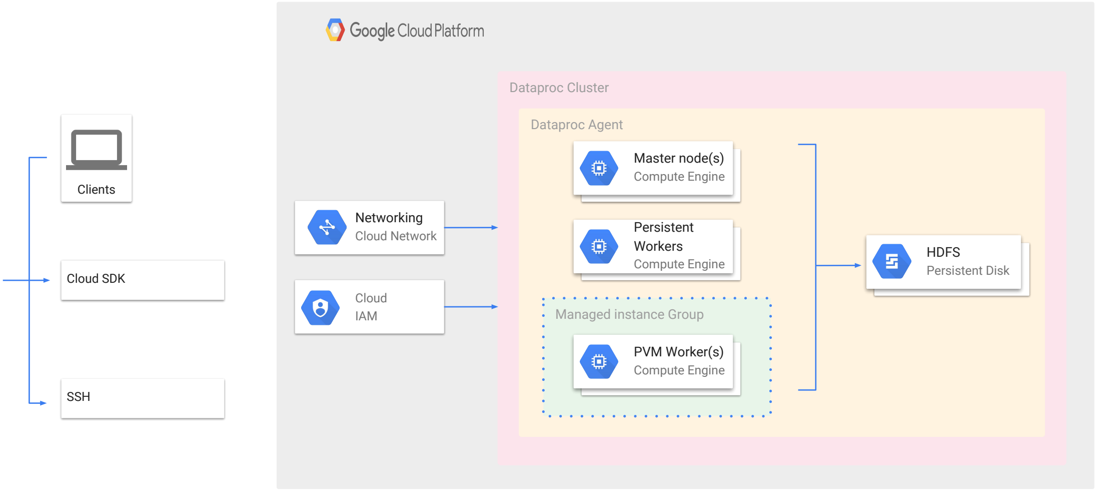
Running jobs + Lab

Separation of storage and compute

Submitting jobs

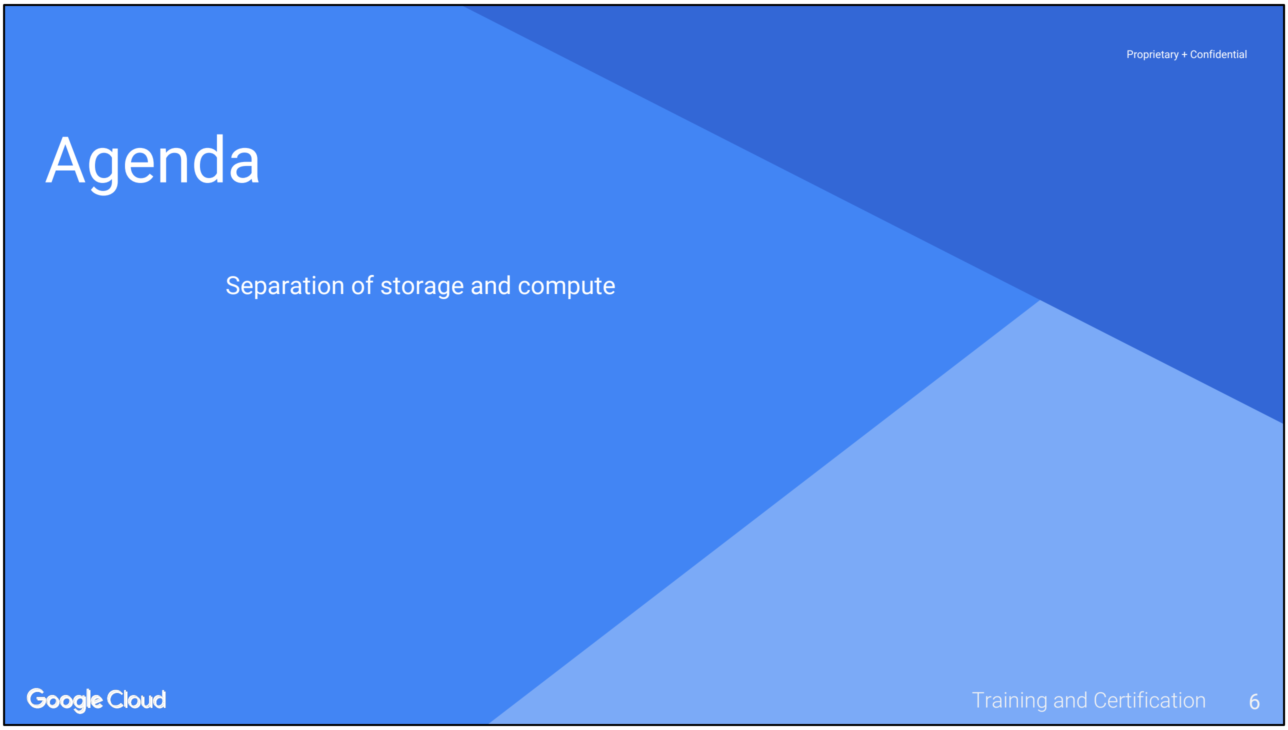
Spark RDDs, Transformations, and Actions + Lab

Can SSH to cluster and run Pig/Spark



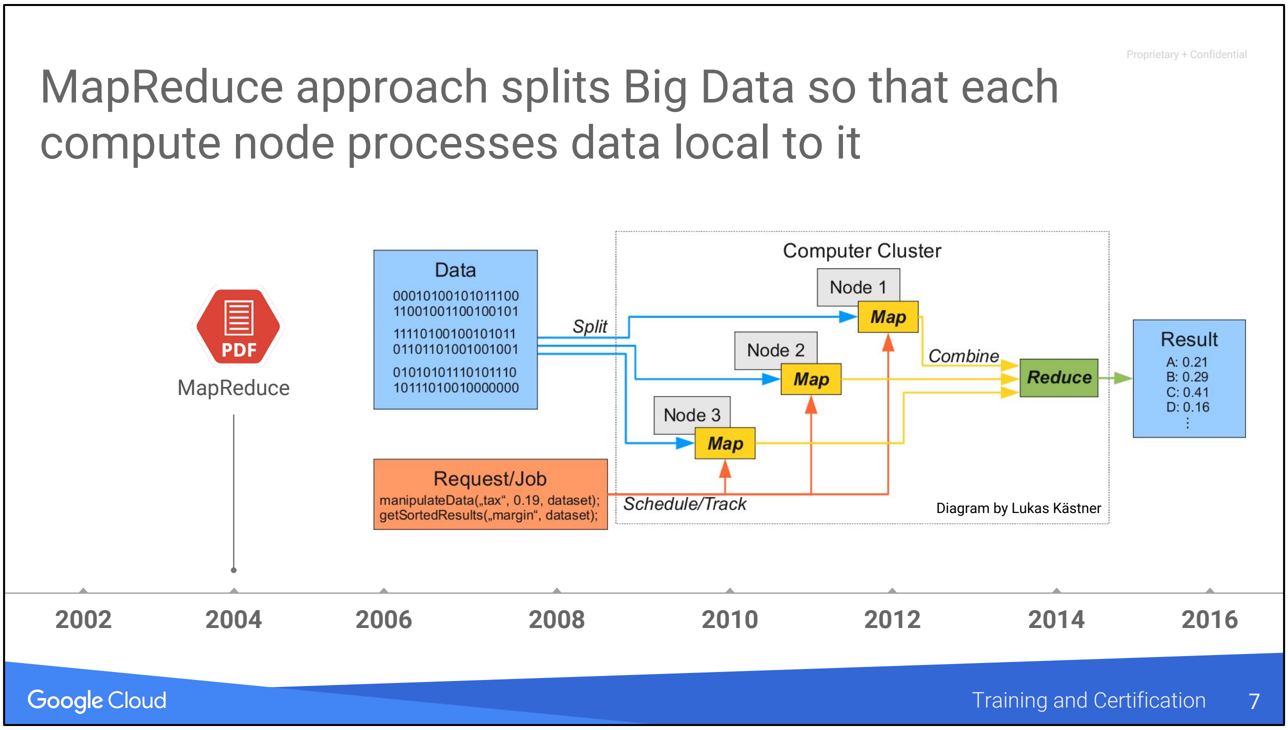
Lab 2: Work with structured and semi-structured data

- Learn about tools for working with structured and semi-structured data
- Use the Hive CLI
- Hive is used for structured data, similar to SQL
- Run a Pig job
- Pig is used for semi-structured data, similar to SQL + scripting



Agenda

Separation of storage and compute



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

PDF

MapReduce

Data

000101001010111100
110010011001001101

111101001001010111
011011010010010001

010101011101011110
101110100100000000

Split

Computer Cluster

Node 1

Map

Node 2

Map

Node 3

Map

Combine

Reduce

Result

A: 0.21
B: 0.29
C: 0.41
D: 0.16
⋮

Request/Job

manipulateData(„tax“, 0.19, dataset);
getSortedResults(„margin“, dataset);

Schedule/Track

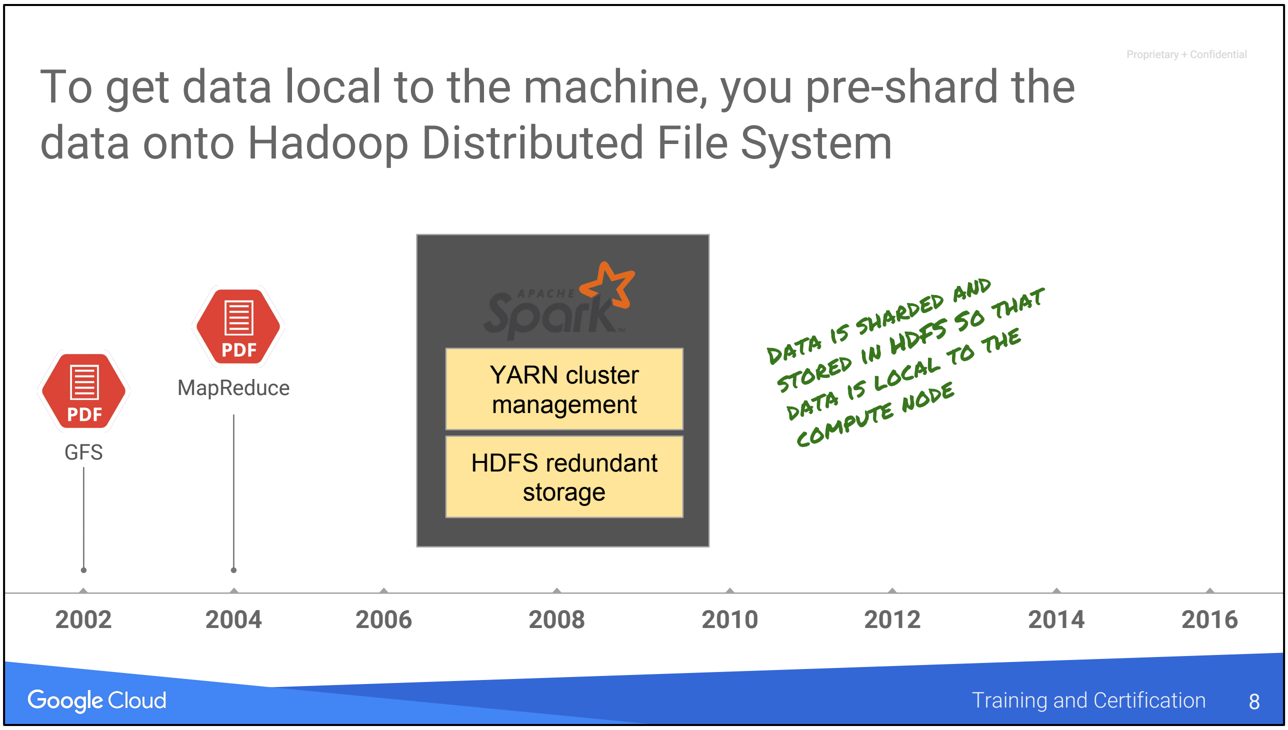
Diagram by Lukas Kästner

2002 2004 2006 2008 2010 2012 2014 2016

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



Proprietary + Confidential

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

PDF GFS

PDF MapReduce

APACHE Spark

YARN cluster management

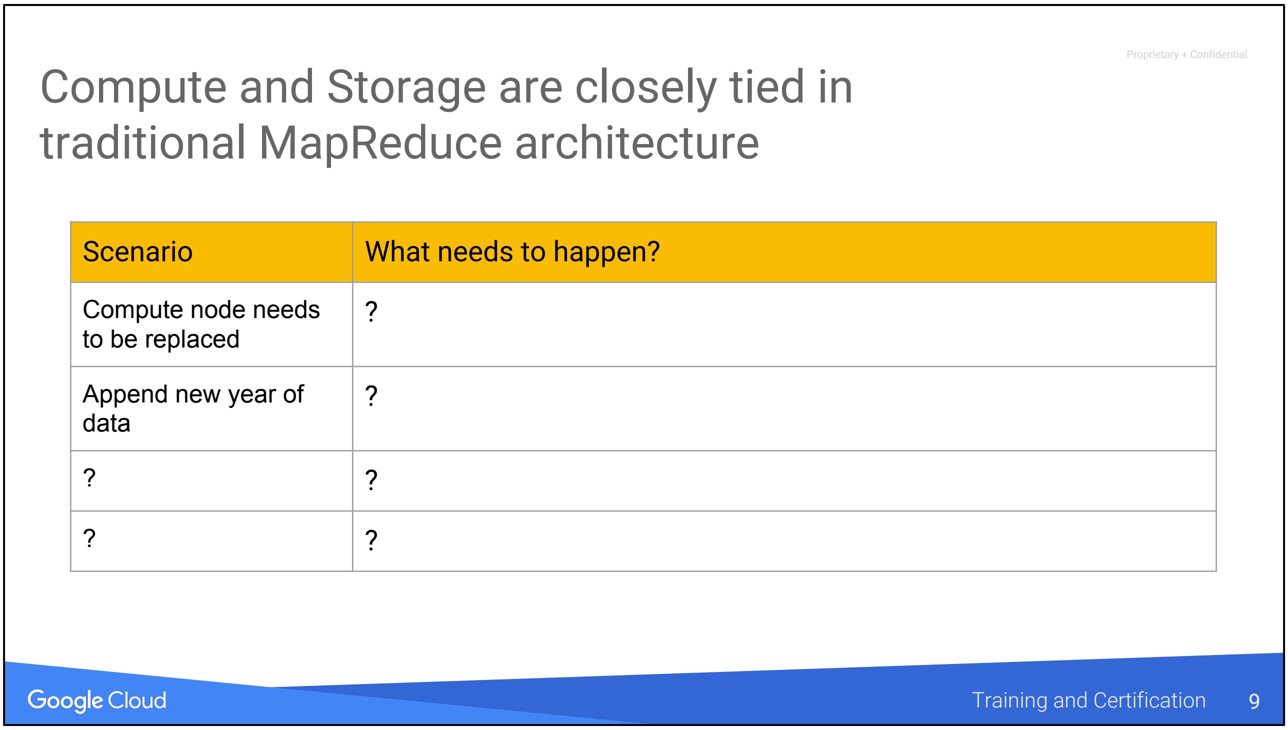
HDFS redundant storage

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?

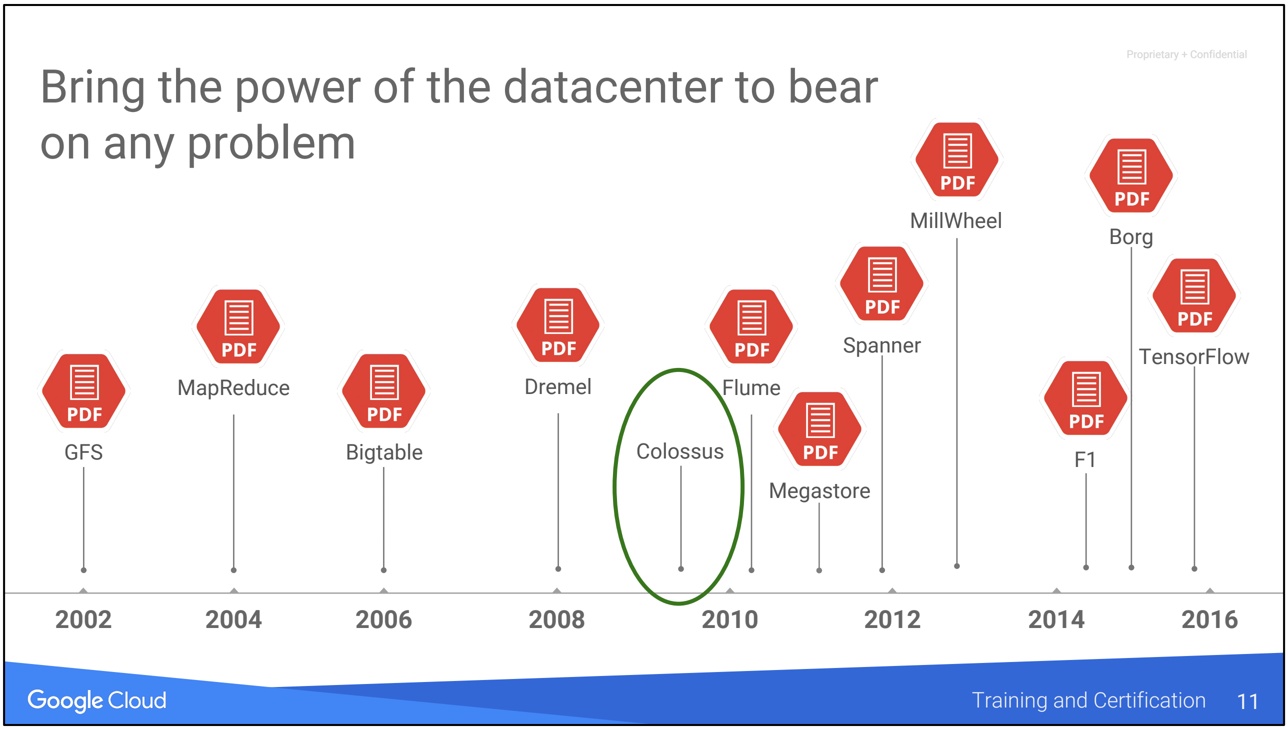
Need to provision increased resources quickly

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

PDF GFS

PDF MapReduce

PDF Bigtable

PDF Dremel

Colossus

PDF Flume

PDF Megastore

PDF Spanner

PDF MillWheel

PDF F1

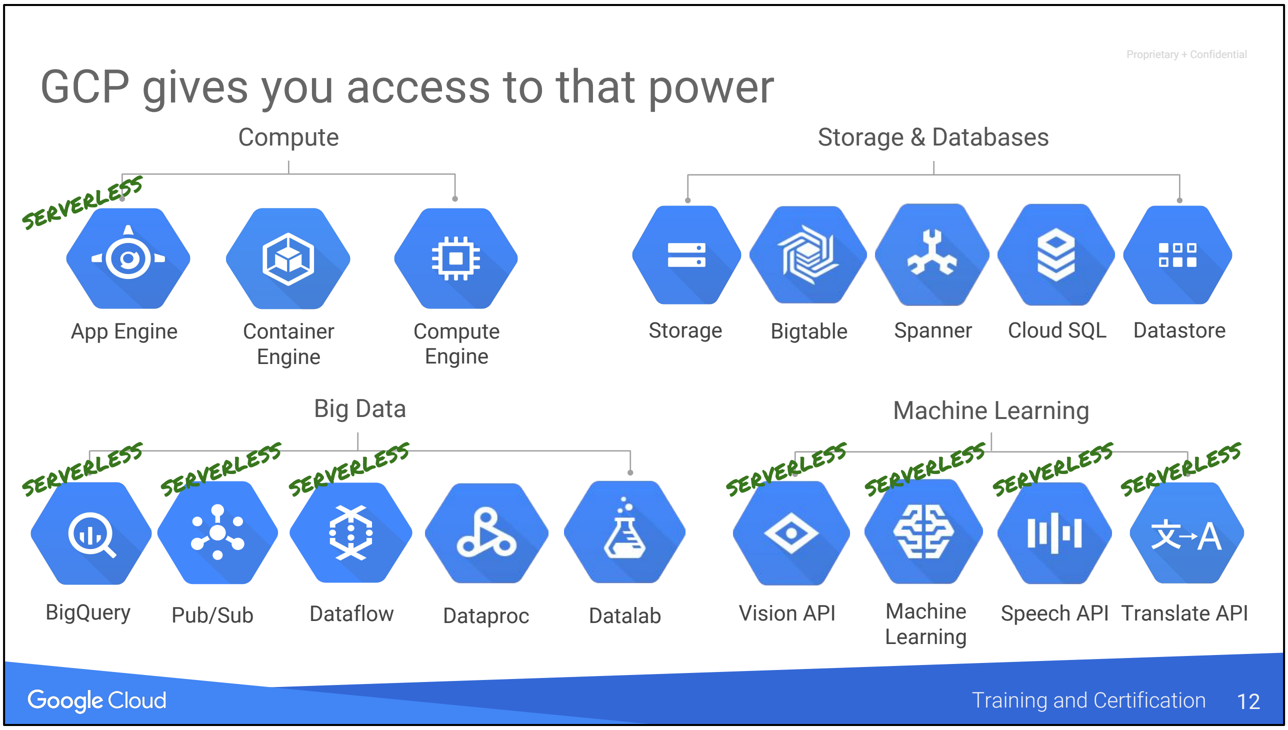
PDF Borg

PDF TensorFlow

2002 2004 2006 2008 2010 2012 2014 2016

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

Compute

- SERVERLESS App Engine
- Container Engine
- Compute Engine

Storage & Databases

- Storage
- Bigtable
- Spanner
- Cloud SQL
- Datastore

Big Data

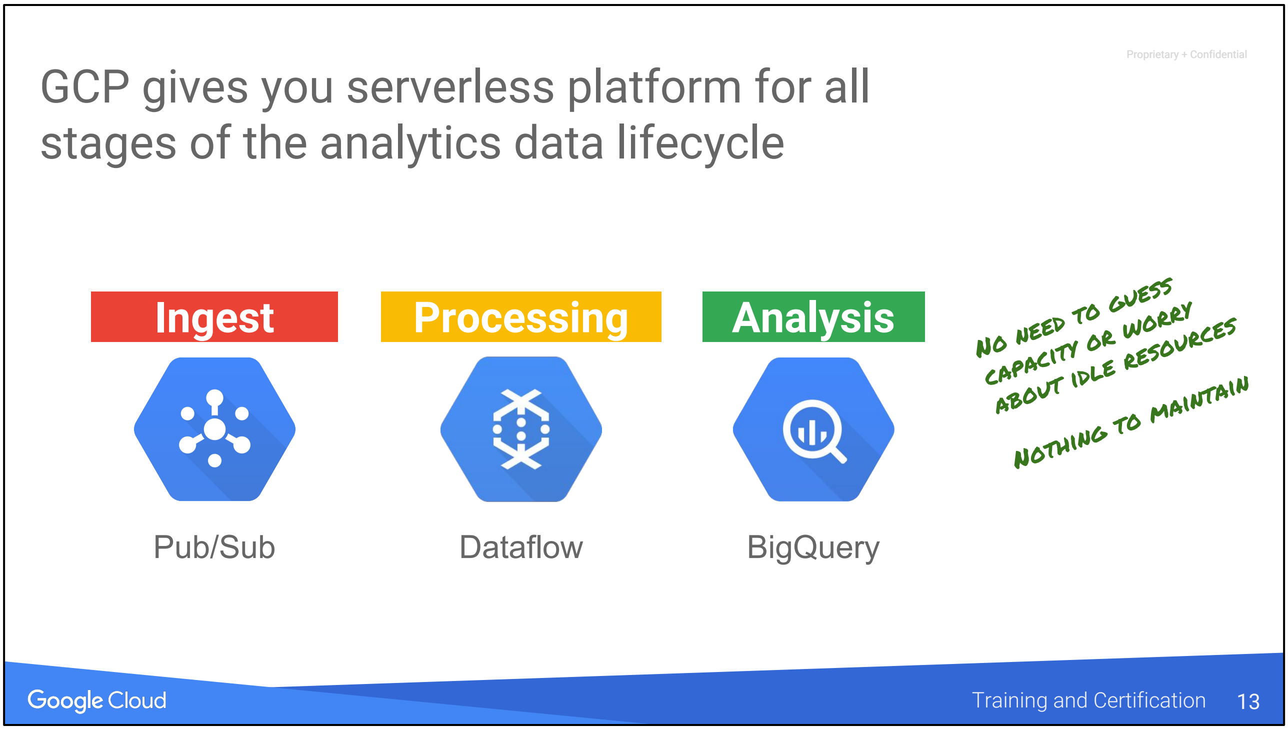
- SERVERLESS BigQuery
- SERVERLESS Pub/Sub
- SERVERLESS Dataflow
- Dataproc
- Datalab

Machine Learning

- SERVERLESS Vision API
- SERVERLESS Machine Learning
- SERVERLESS Speech API
- SERVERLESS Translate API

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

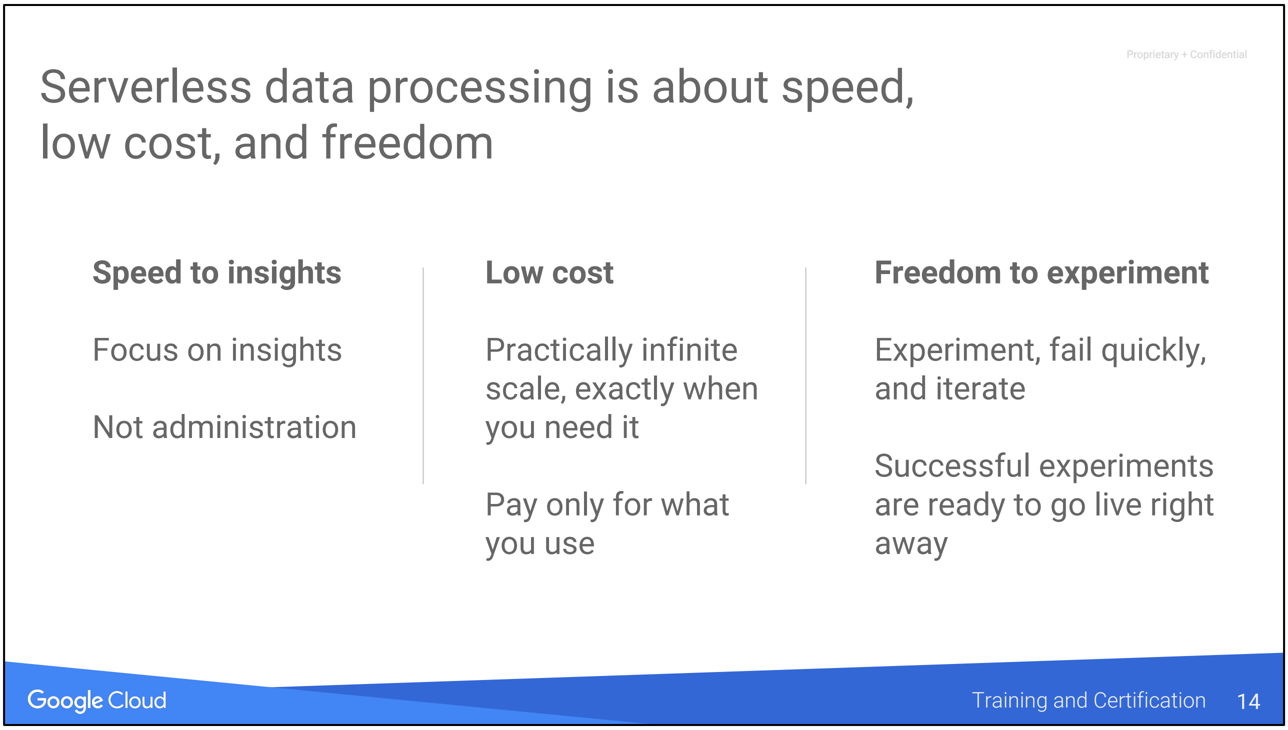
Ingest	Processing	Analysis
Pub/Sub	Dataflow	BigQuery

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

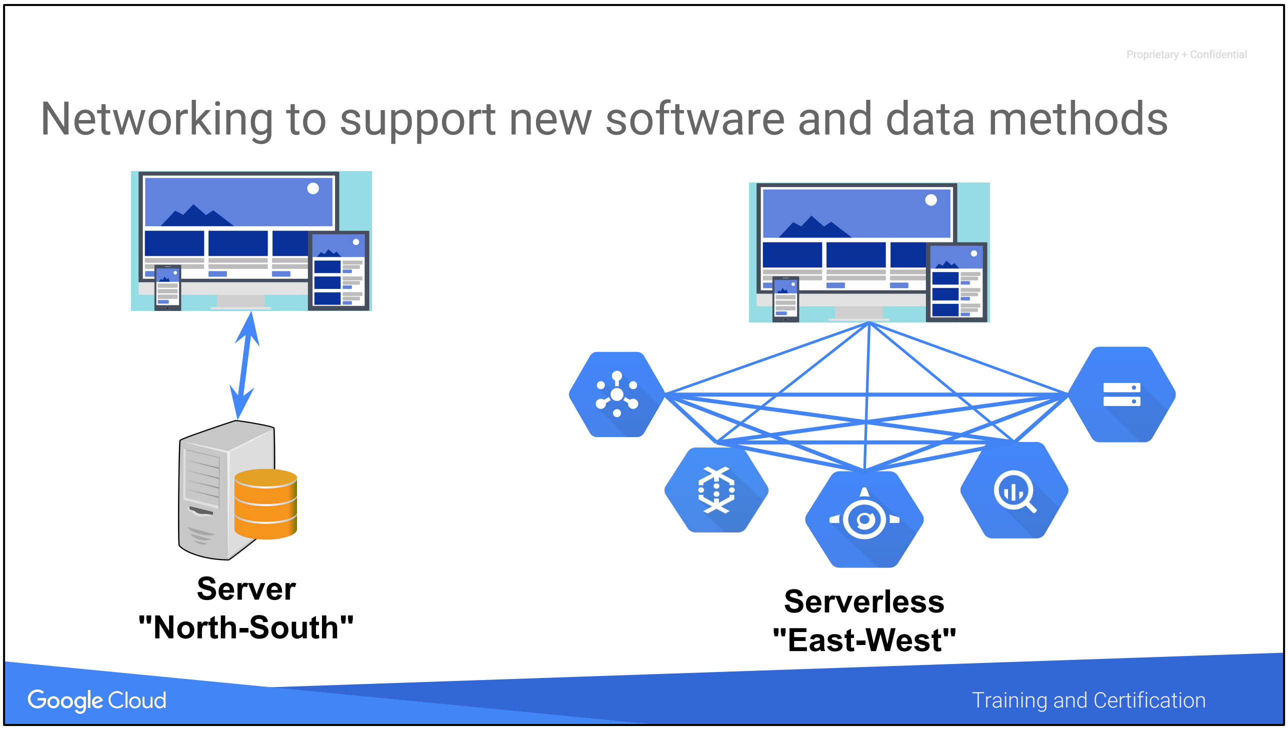
Freedom to experiment

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.



Networking to support new software and data methods

Server "North-South"

Serverless "East-West"

Google Cloud

Training and Certification

Networking within a data center used to be focused on transactions between a client and a server, what is called "North-South" communications. For example, a user might request a web page, and the server generates the web page. In that paradigm, the network needs to support low latency transactions between the user and the server.

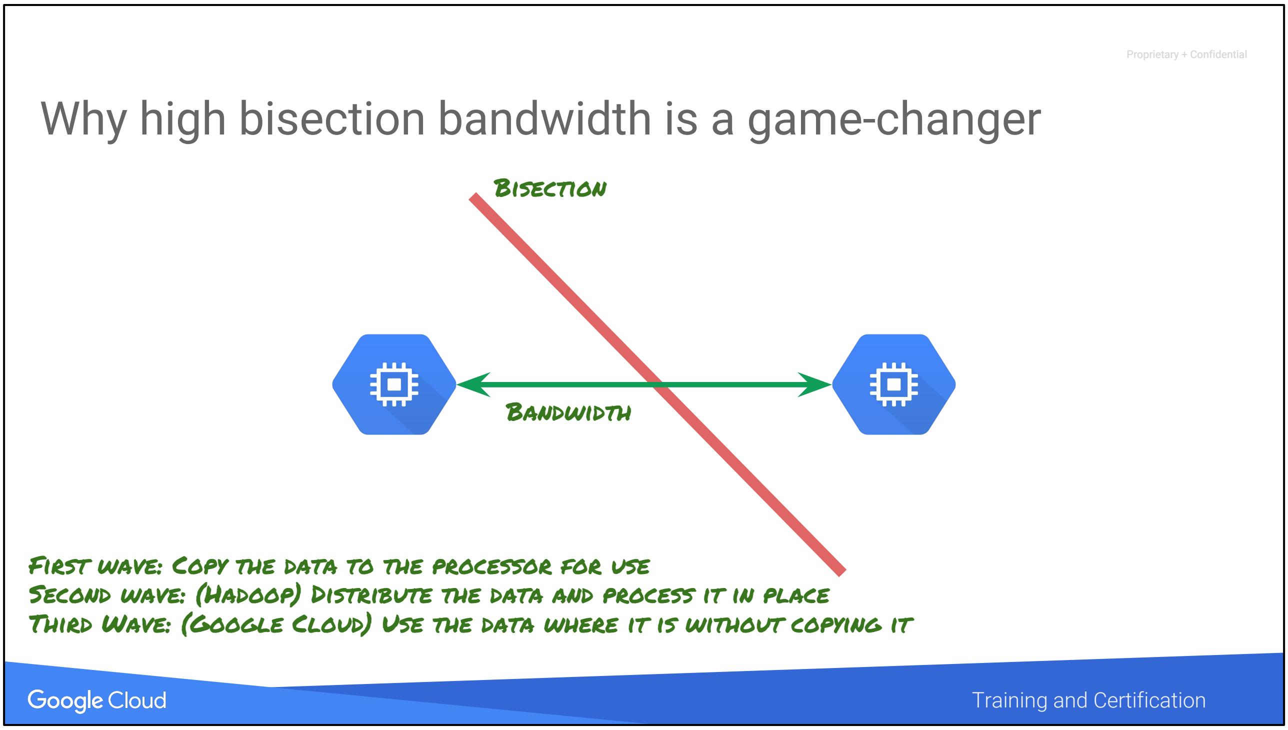
Networking for cloud applications must account for distributed data and "serverless" services. (Serverless, of course, meaning that the servers, if they exist, are not exposed to the client/user. The client only has visibility to an API, not to the VMs.) In the web page example, the single web page might be built from data drawn from many services. Beneath those services might be communications between thousands of servers. So very fast server-to-server communication inside the network becomes a most important network design criteria. This is called "East-West" communications.

One way of measuring this or describing "East-West" communications is "bisectional bandwidth".

Google Cloud's internal network supports "petabit bisection bandwidth".

More on this subject:
http://highscalability.com/blog/2015/8/10/how-google-invented-an-amazing-datacenter-network-only-they.html

https://pixabay.com/en/server-web-network-data-computer-567944/
https://pixabay.com/en/responsive-web-pages-websites-1622825/



Proprietary + Confidential

Why high bisection bandwidth is a game-changer

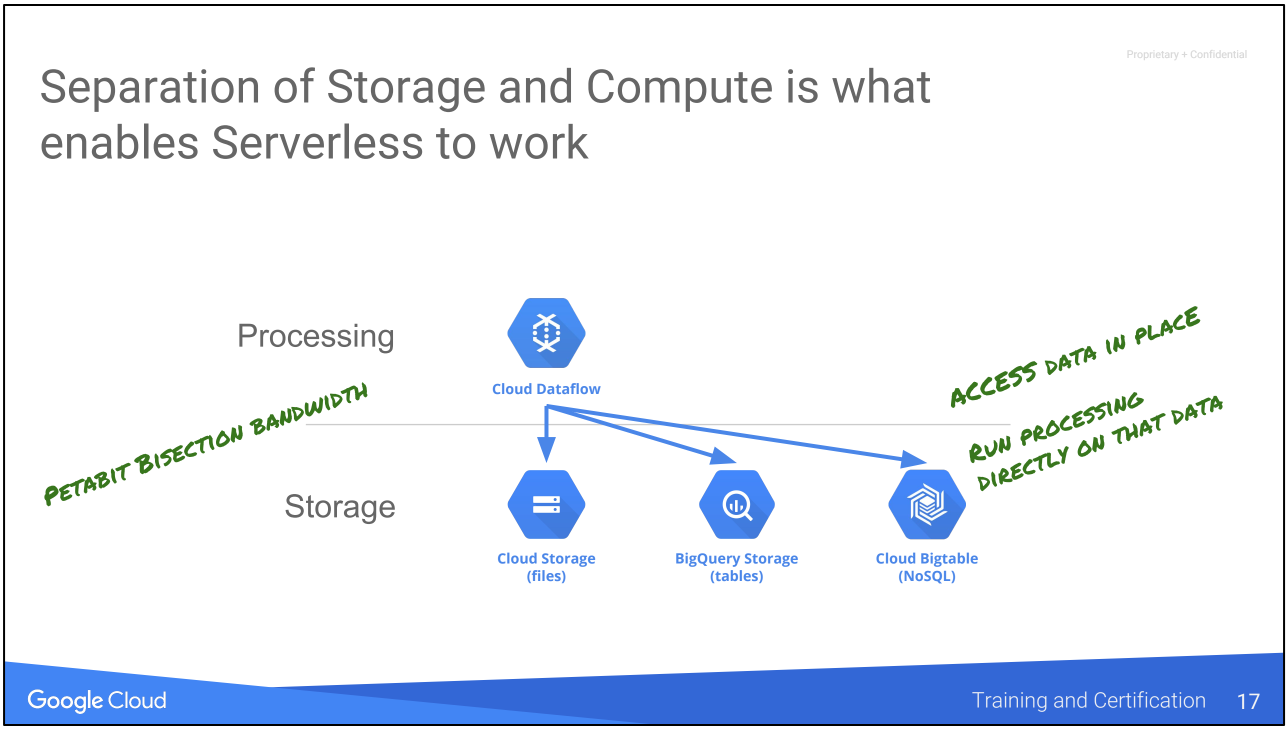
FIRST WAVE: COPY THE DATA TO THE PROCESSOR FOR USE
SECOND WAVE: (HADOOP) DISTRIBUTE THE DATA AND PROCESS IT IN PLACE
THIRD WAVE: (GOOGLE CLOUD) USE THE DATA WHERE IT IS WITHOUT COPYING IT

Google Cloud

Training and Certification

Bisectional bandwidth

If you draw a line somewhere in a network, bisectional bandwidth is the rate of communication at which servers on one side of the line can communicate with servers on the other side. With enough bisectional bandwidth any server can communicate with any other server at full network speeds. With petabit bisectional bandwidth, the communication is so fast that it no longer makes sense to transfer files and store them locally. Instead, it makes sense to use the data from where it is stored.



Proprietary + Confidential

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

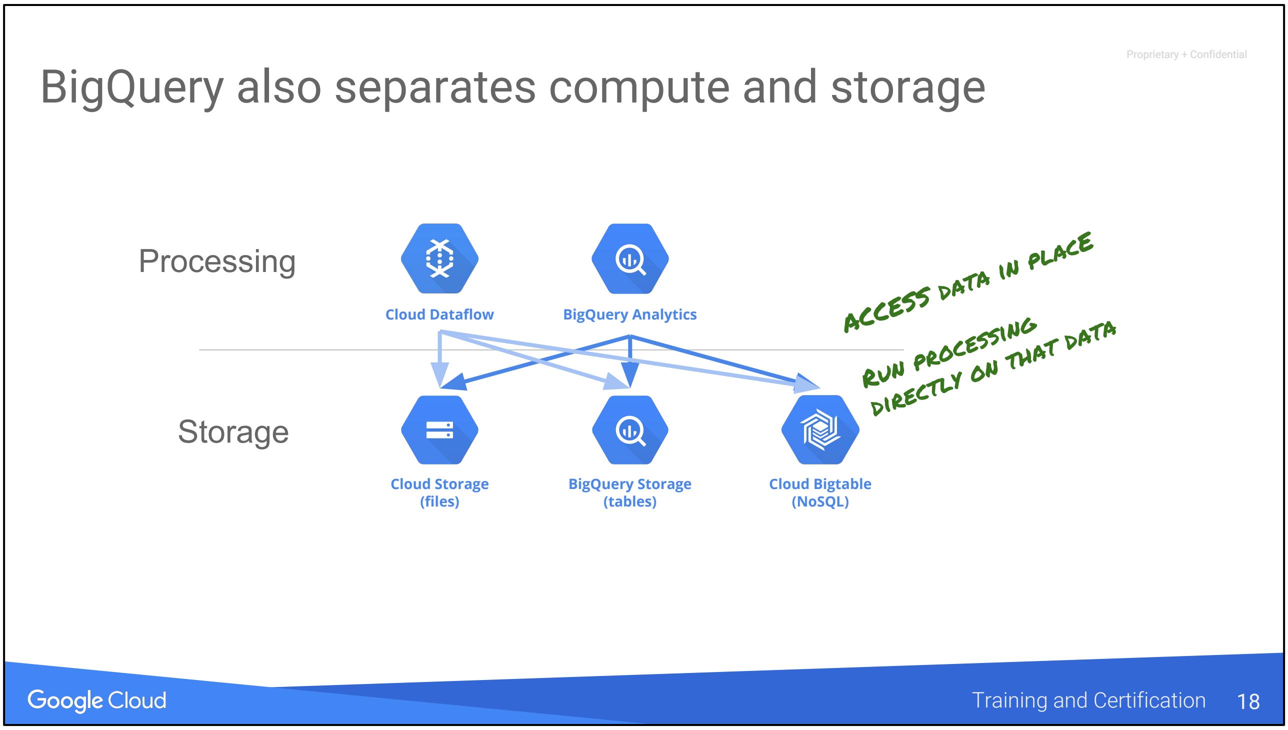
RUN PROCESSING DIRECTLY ON THAT DATA

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.



BigQuery also separates compute and storage

ACCESS DATA IN PLACE

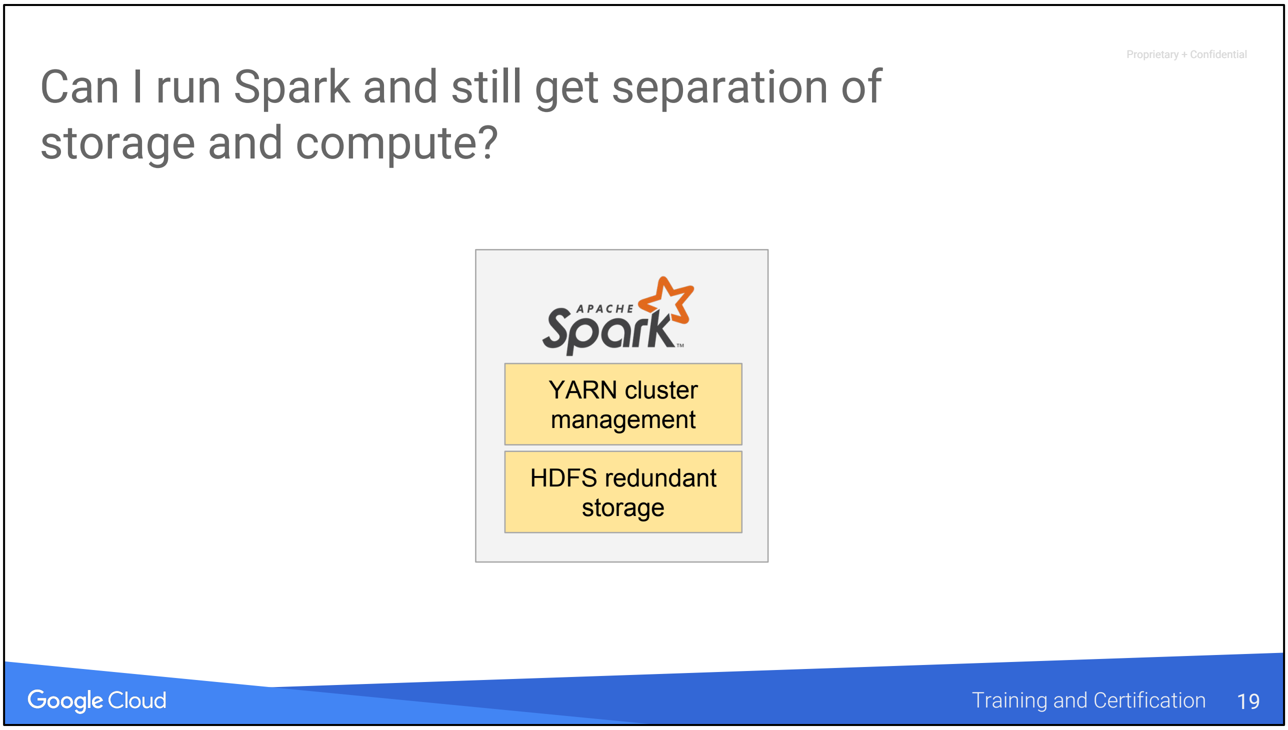
RUN PROCESSING DIRECTLY ON THAT DATA

Notes:

Access any storage system from any processing tool.

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

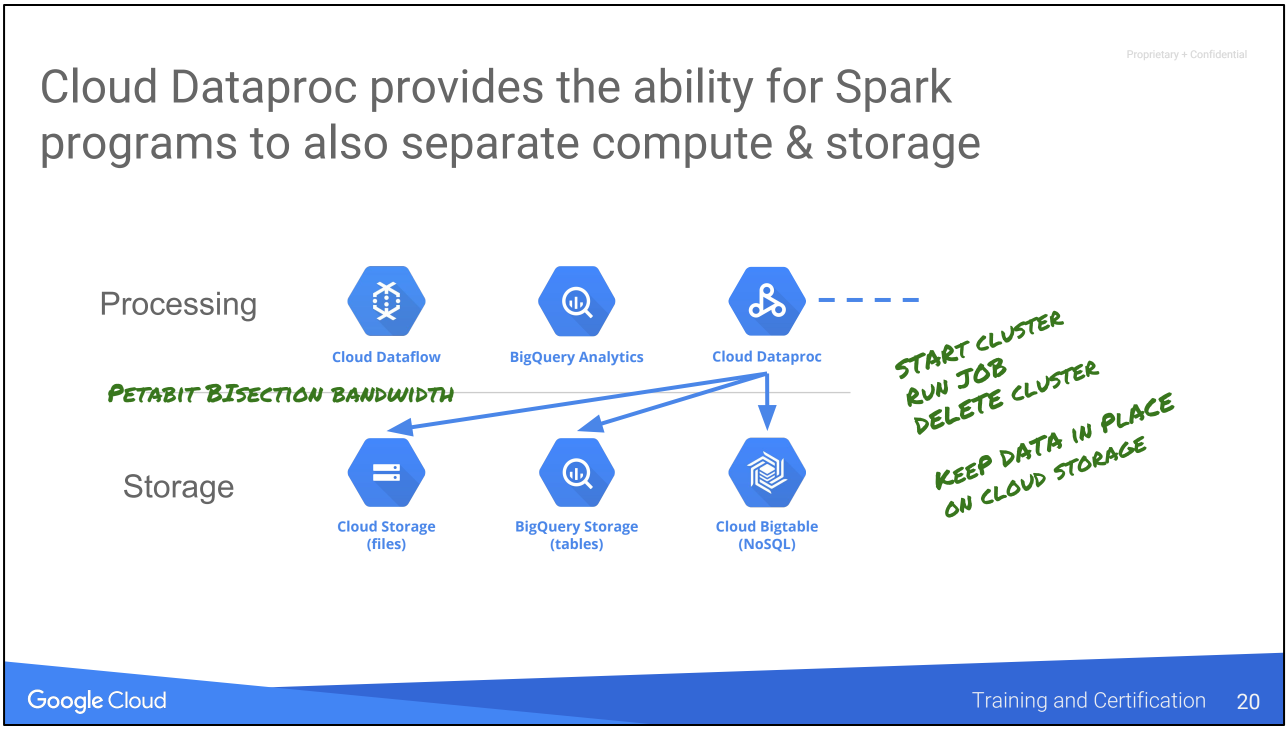
BigQuery is "just" a query engine. It can query csv files on cloud storage also, for example.



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

START CLUSTER
RUN JOB
DELETE CLUSTER

KEEP DATA IN PLACE
ON CLOUD STORAGE

Storage

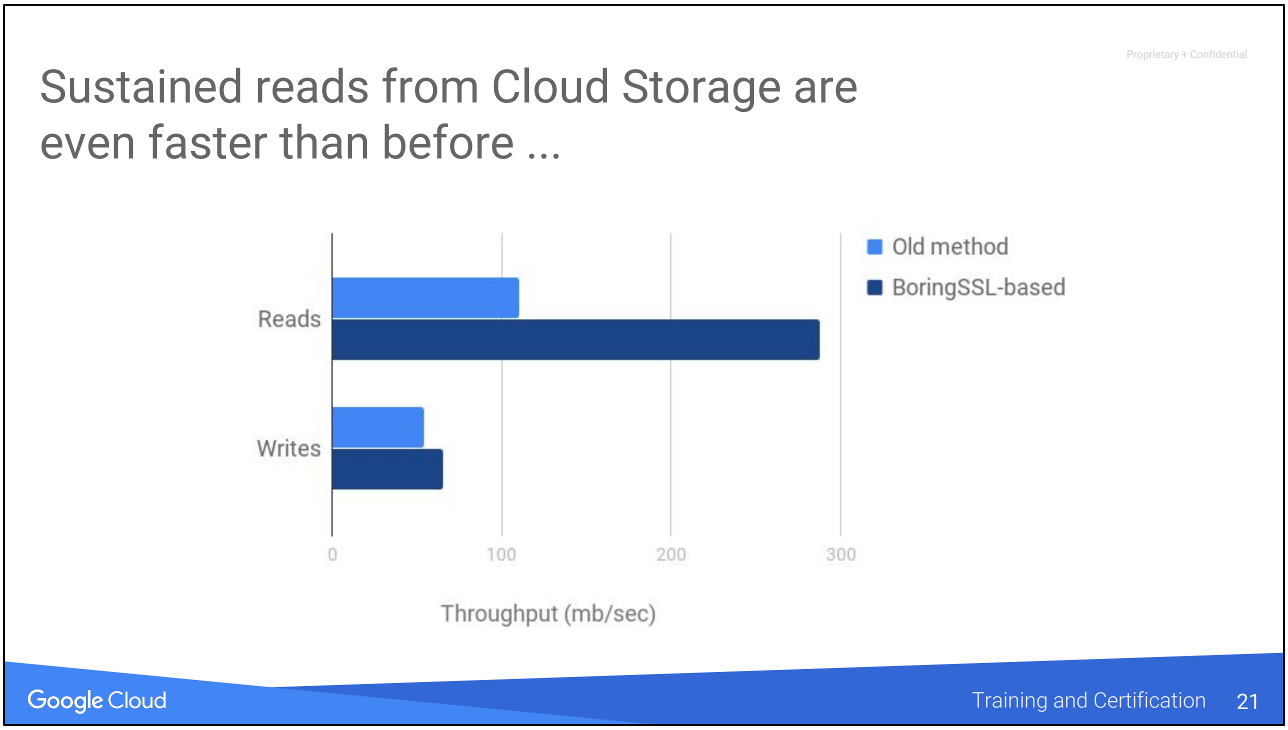
Cloud Storage
(files)

BigQuery Storage
(tables)

Cloud Bigtable
(NoSQL)

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

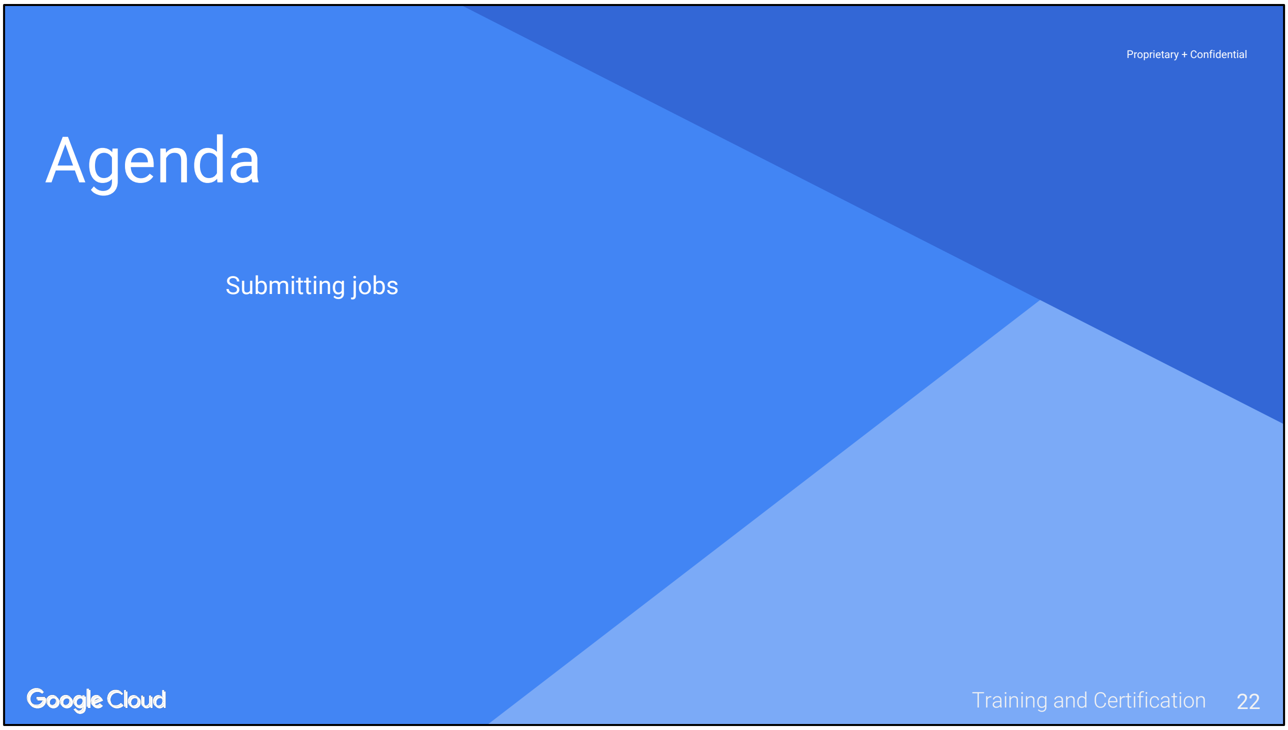


Sustained reads from Cloud Storage are even faster than before ...

Notes:

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

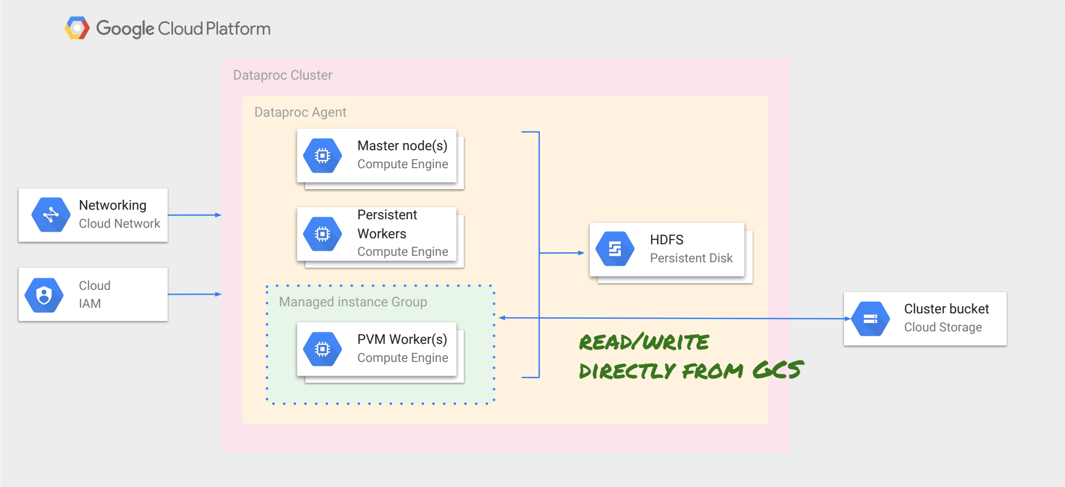
Old: 2016
New: 2017

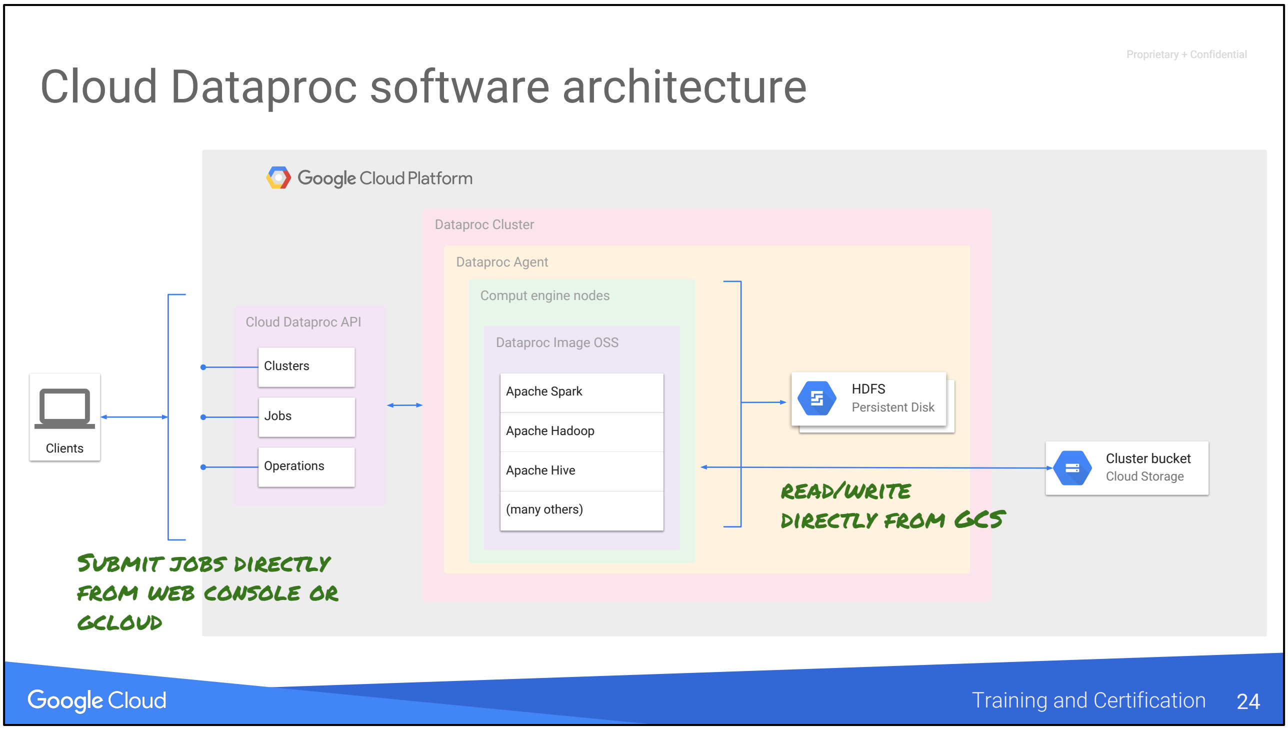


Agenda

Submitting jobs

Cloud Dataproc hardware architecture





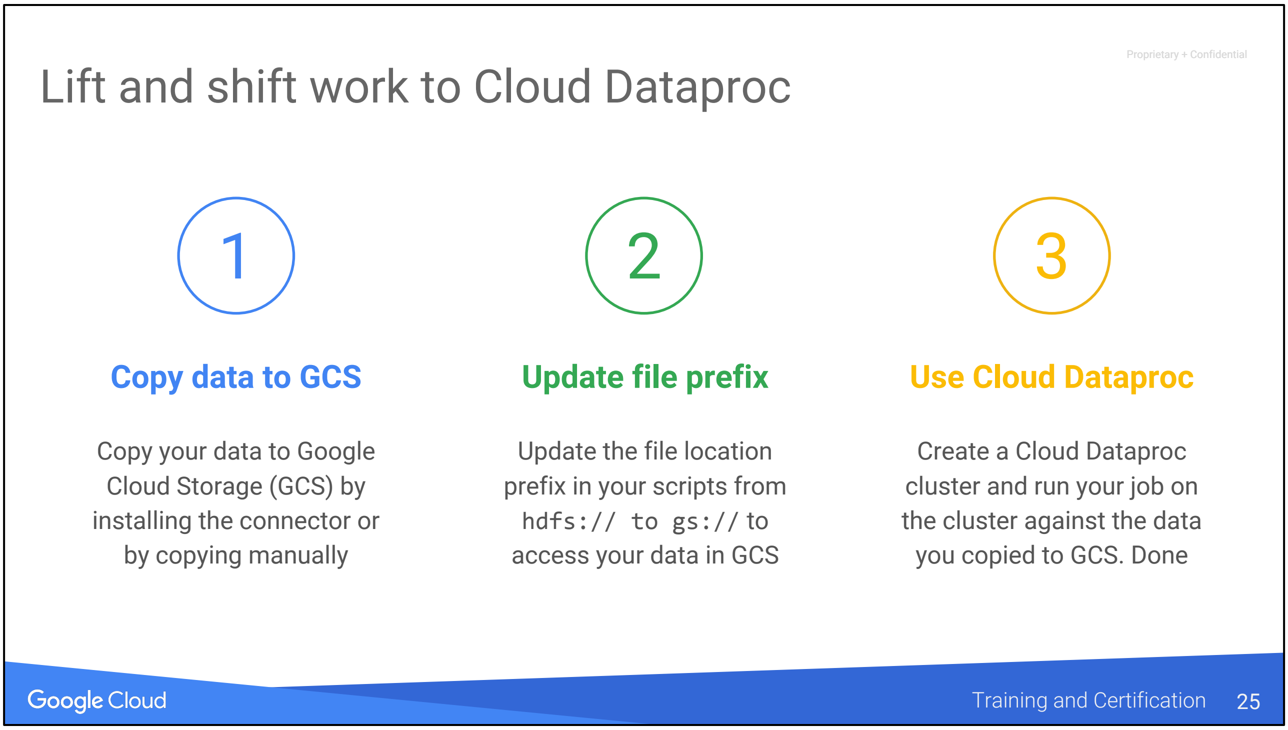
Cloud Dataproc software architecture

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

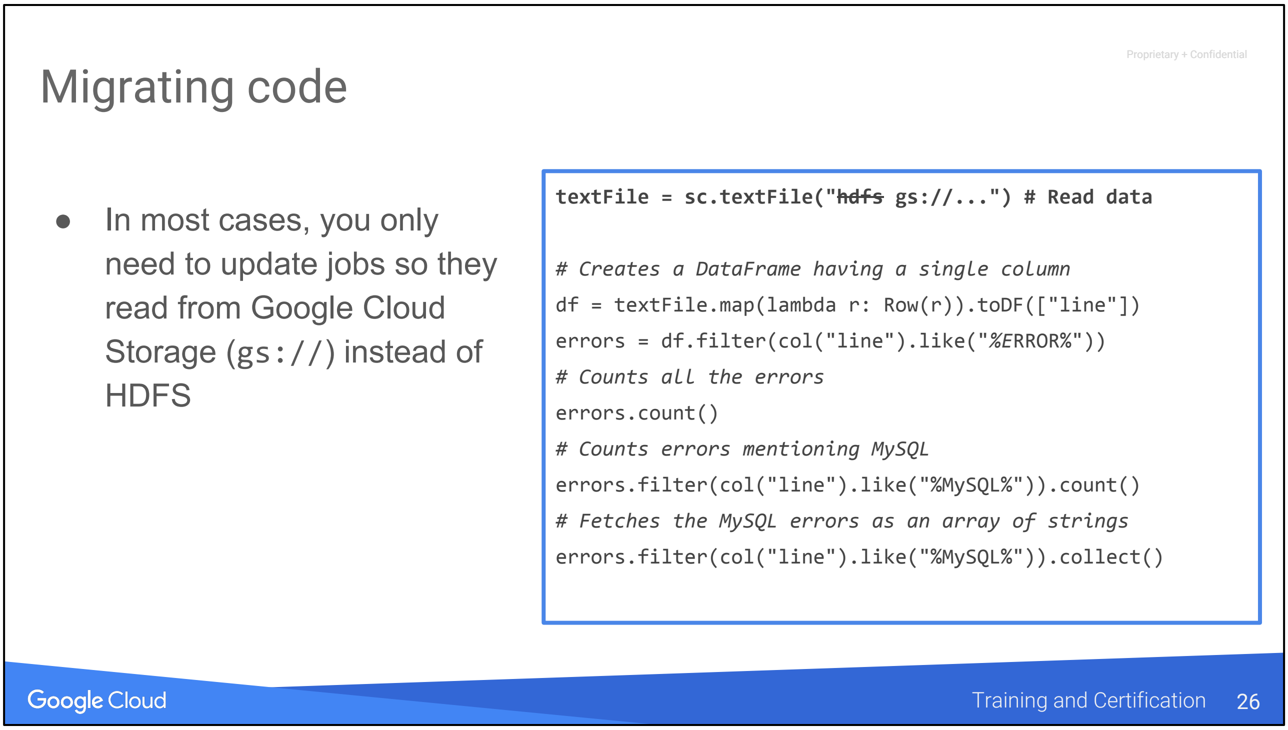
Update file prefix

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

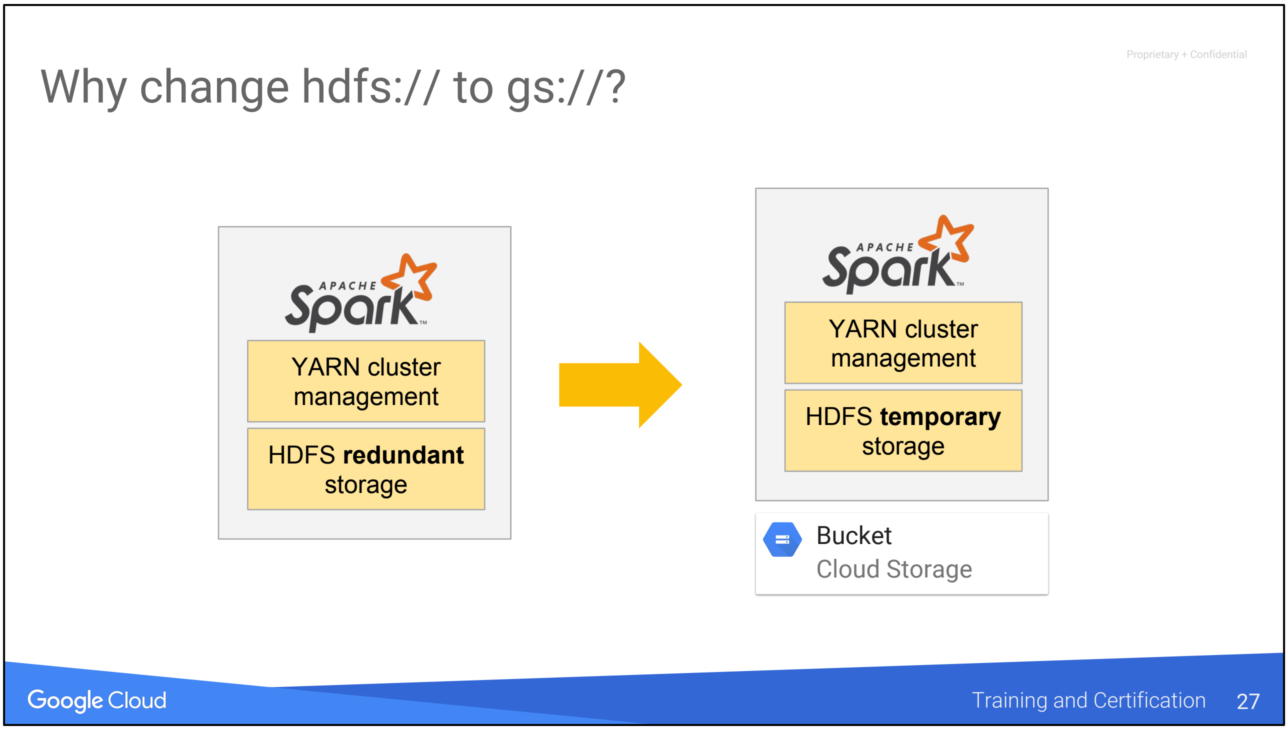


Migrating code

- 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://?

YARN cluster management

HDFS **redundant** storage

YARN cluster management

HDFS **temporary** storage

Bucket
Cloud Storage

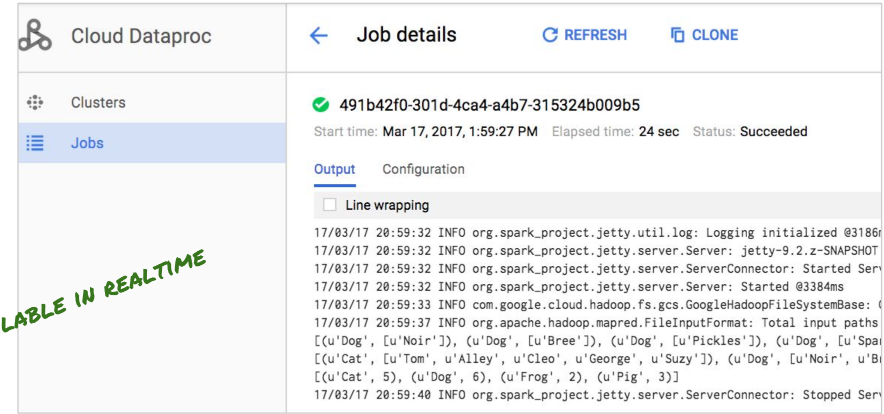
Notes:

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

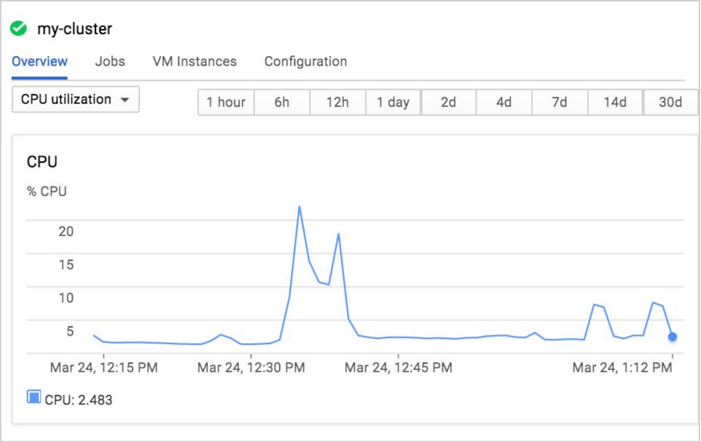
```
17/03/17 20:59:32 INFO org.spark_project.jetty.util.log: Logging initialized @3186
17/03/17 20:59:32 INFO org.spark_project.jetty.server.Server: jetty-9.2.z-SNAPSHOT
17/03/17 20:59:32 INFO org.spark_project.jetty.server.ServerConnector: Started Ser
17/03/17 20:59:32 INFO org.spark_project.jetty.server.Server: 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



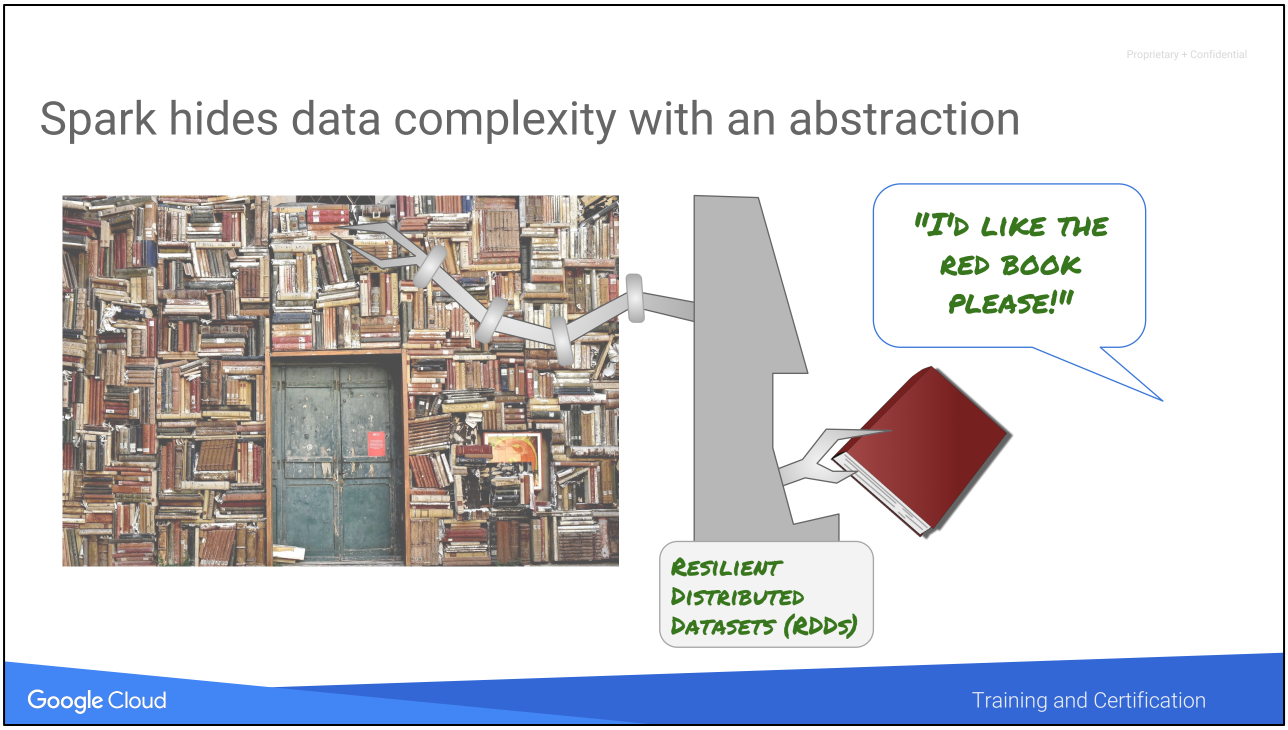
Notes:

Overall cluster usage from Dataproc page.

Individual VMs from Compute Engine VM.

Agenda

Spark RDDs, Transformations, and Actions + Lab

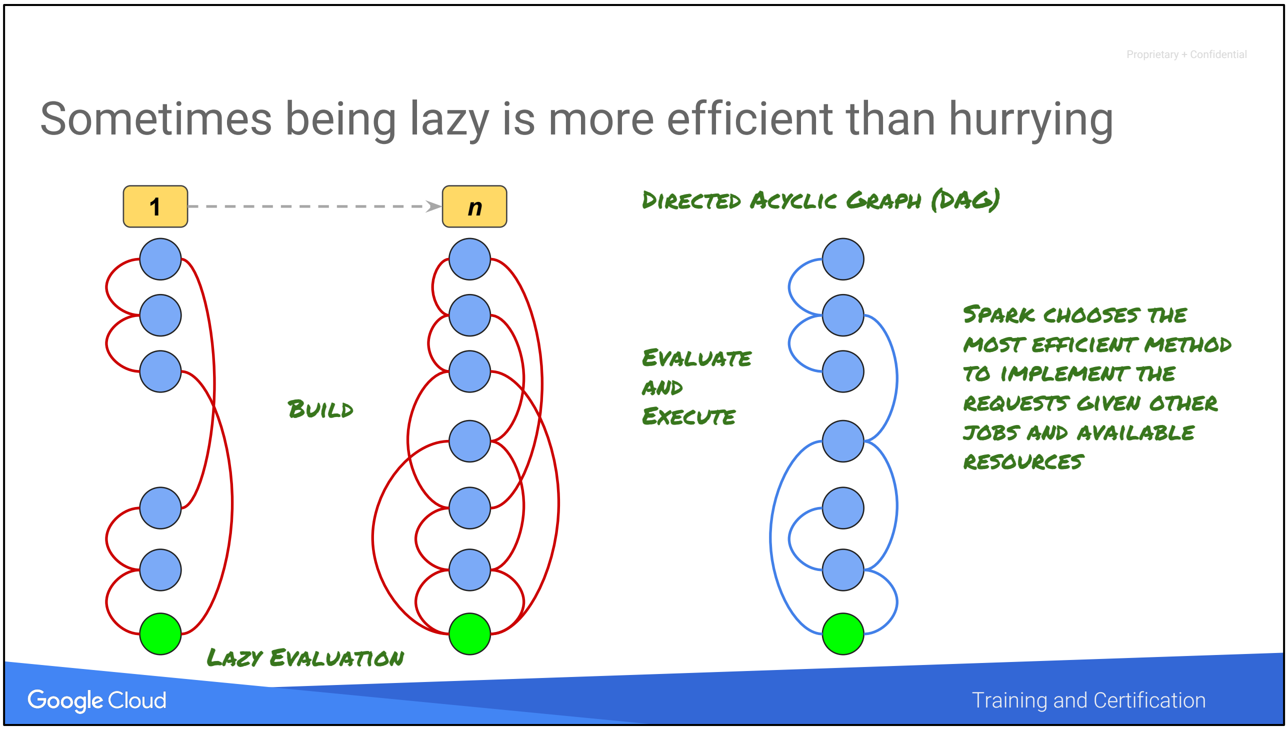


Spark hides data complexity with an abstraction

"I'D LIKE THE RED BOOK PLEASE!"
RESILIENT DISTRIBUTED DATASETS (RDDS)

RDDs hide the complexity of the location of data within the cluster, and also the complexity of replication.
Spark partitions data in memory across the cluster and knows how to recover through an RDD's lineage, should anything go wrong.
Spark has the ability to direct processing to occur where there is processing resource available.
You treat your data as a single entity, Spark knows the truth.
Data partitioning, Data replication, Data recovery, Pipelining of processing -- all are automated by Spark so you don't have to worry about them.

https://pixabay.com/en/books-door-entrance-italy-colors-1655783/
https://pixabay.com/en/book-red-closed-library-education-34014/



Proprietary + Confidential

Sometimes being lazy is more efficient than hurrying

Directed Acyclic Graph (DAG)

Build

Lazy Evaluation

Evaluate and Execute

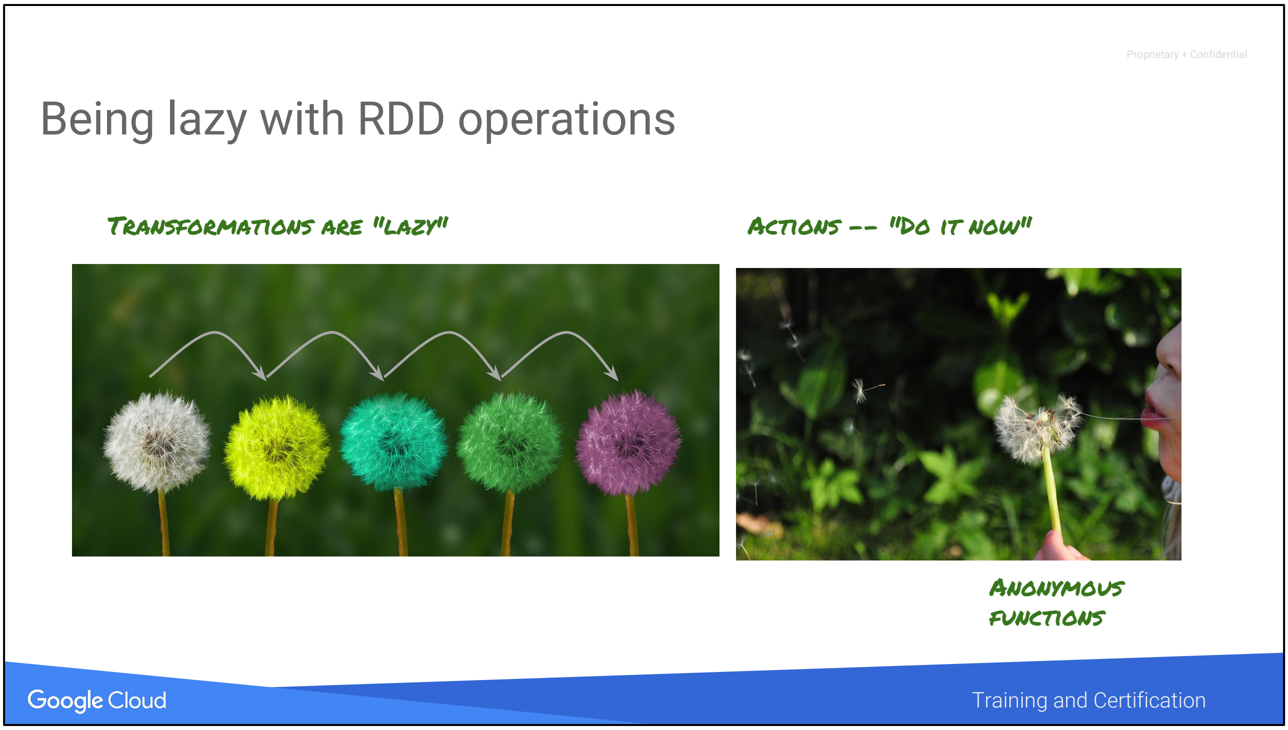
Spark chooses the most efficient method to implement the requests given other jobs and available resources

Google Cloud

Training and Certification

When you program in command languages, you "tell the system what to do".
With Spark, you program with "requests". Spark doesn't immediately perform these actions. Instead, it stores them in a graph system called a DAG. Only when a request is submitted that requires output, does Spark actually process the data.

The benefit of this strategy is that Spark can look at all the requests and the intermediate results, and construct parallel "pipelines" based on the resources that are available in the cluster at that time. This hides a lot of complexity from the user of the service. It also allows Spark to mix different kinds of applications -- those that are more processing intensive and those that are more data intensive -- and balance the work flows. Before Spark (with MapReduce) the cluster has to be "tuned" for the kinds of applications being run.



Proprietary + Confidential

Being lazy with RDD operations

TRANSFORMATIONS ARE "LAZY"

ACTIONS -- "DO IT NOW"

ANONYMOUS FUNCTIONS

Google Cloud

Training and Certification

Transformation
Input is an RDD and output is an RDD
Registered in DAG awaiting an action (lazy)

Action
Output is a result format, such as a text file
Triggers Spark to process the pipeline

Transformations and Actions are API calls that reference the functions you want them to perform.

Anonymous functions (in Python, Lambda functions) are commonly used for the following reasons:

- A self-contained way to make a request to Spark
- Lambda functions are defined "inline" making it easy to read and understand in sequence.
- They are limited to a single specific purpose.
- They don't clutter the namespace with function names for code that is only used in one place.

https://pixabay.com/en/dandelion-colorful-people-of-color-2817950/

https://pixabay.com/en/blowball-dandelion-girl-blowing-384598/

Lab 3: Work with unstructured data; Submit Dataproc Jobs; Explore Spark

- Work with unstructured data
- Submit Dataproc Jobs
- Explore Spark
- Explore HDFS and Cloud Storage
- Use interactive PySpark to learn about RDDs
- Learn about RDD operations (transformation and actions)

cloud.google.com

Images by Connie Zhou