

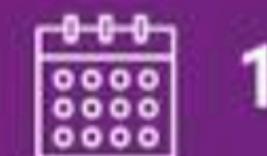
# Making Data Pipelines easy

With Dataproc and Composer

Julian Joseph, Feb 15, 2023

## DataHour

# Making Data Pipelines easy with Dataproc and Composer



17 February 2023



8:30 PM - 9:30 PM IST

**Julian Sara Joseph**Developer Advocate:  
Data Analytics, Data Science

Google Cloud

## DataHour: Making Data Pipelines Easy with Dataproc and Composer



Online



17-02-2023 08:30 PM to 17-02-2023 09:30 PM



4303

Registered



Knowledge and Learning.

Prizes

— STARTS IN —

16 33 5

HOURS MIN SEC

[REGISTER](#)

# In this session

## Topics

- Dataproc
  - Personal Story
  - Intro to Dataproc
  - Serverless Spark
  - Demo on Serverless Spark
- Composer
  - Dags and operators
  - Demo - Composer for Dataproc Serverless workloads
- Q&A

# A walk down memory lane

## Startup

- 2016
  - Data Engineer in a small product team
  - Our client was in retail and the Data was in TBs and it was sent weekly.
  - ETL - It was our job to create deltas and make aggregates before storing the data for our DS team
- 2018
  - 2nd Client - Data was in PBs
  - Our spark jobs needed varying levels of compute power
  - Still manually triggering cron jobs weekly to start the spark jobs

# Cron jobs

```
GNU nano 4.8                               /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * user-name command to be executed
17 *      * * *    root     cd / && run-parts --report /etc/cron.hourly
25 6      * * *    root     test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6      * * 7    root     test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6      1 * *    root     test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly)>

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell  ^  Go To Line M-E Redo
```

# We needed

Break down business problems into actionable items

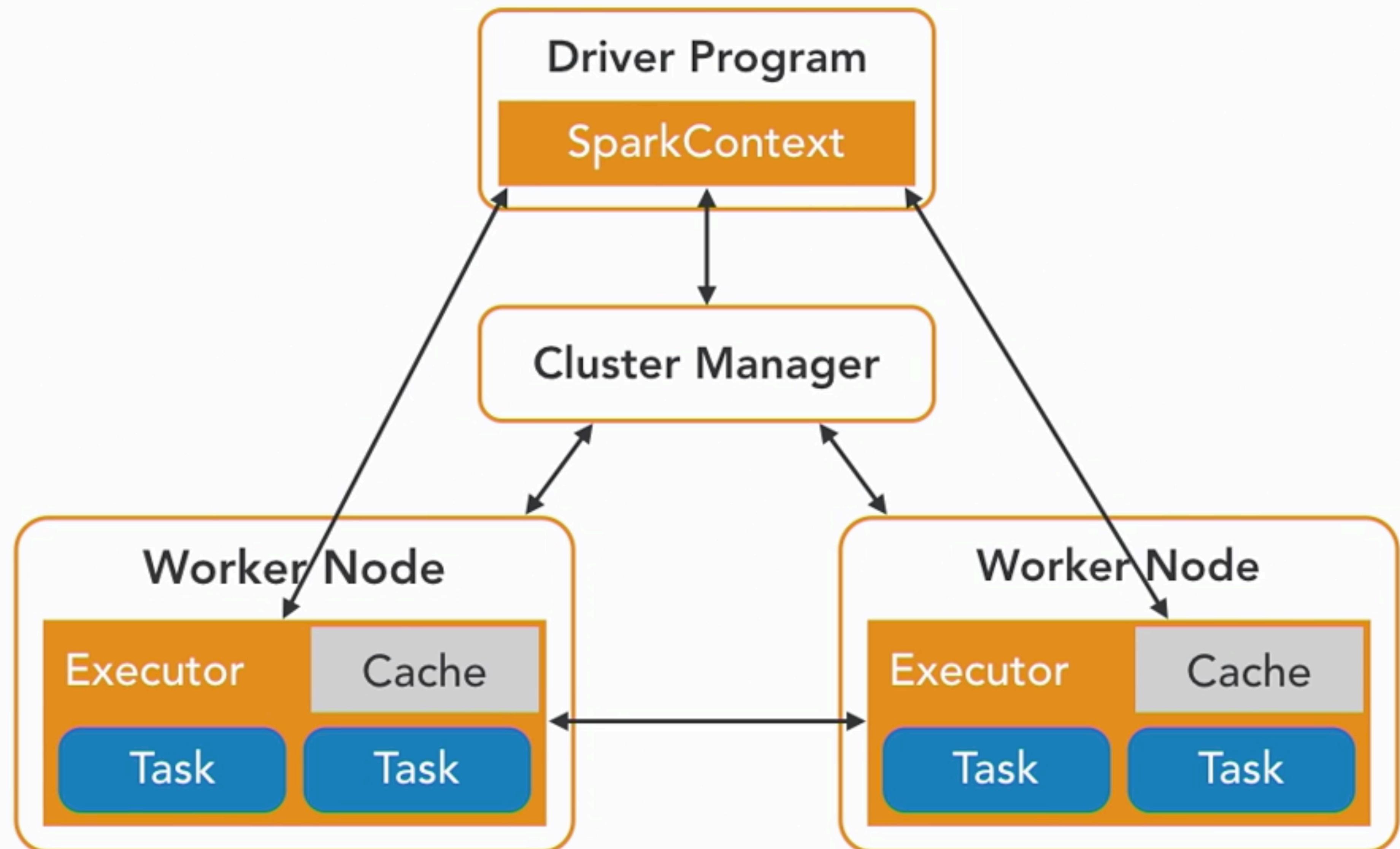
- Auto scaling clusters/nodes + Cost effective (Small budget team) - ??
- Automation of our cron jobs -??

# Why Spark

Data Pipelines were working just fine with HDFS right?

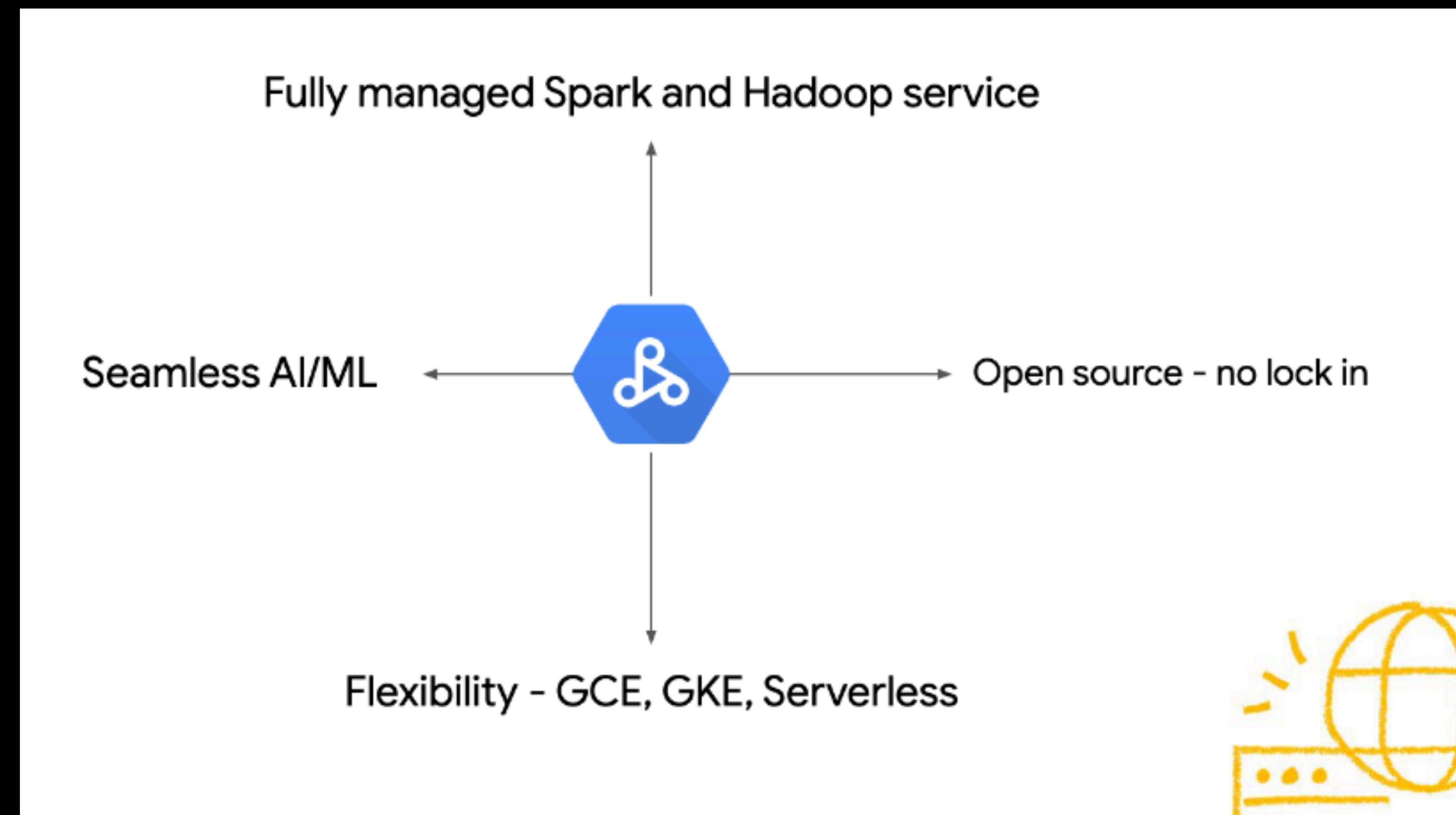
- In 2004 - MapReduce Algorithm was released
- In 2006 - Hadoop based on MapReduce was released
- In 2009 - UC Berkely developed Spark - Given to Apache Software Foundation
- Chief difference - Spark and MapReduce - Spark processes and keeps data in memory for all subsequent steps - without writing to or reading from disk!

# Spark Components



# Enter Dataproc on Google Cloud

Or EMR on AWS



**Demo time - Dataproc Cluster Intro - Swipe**

**Low Cost**

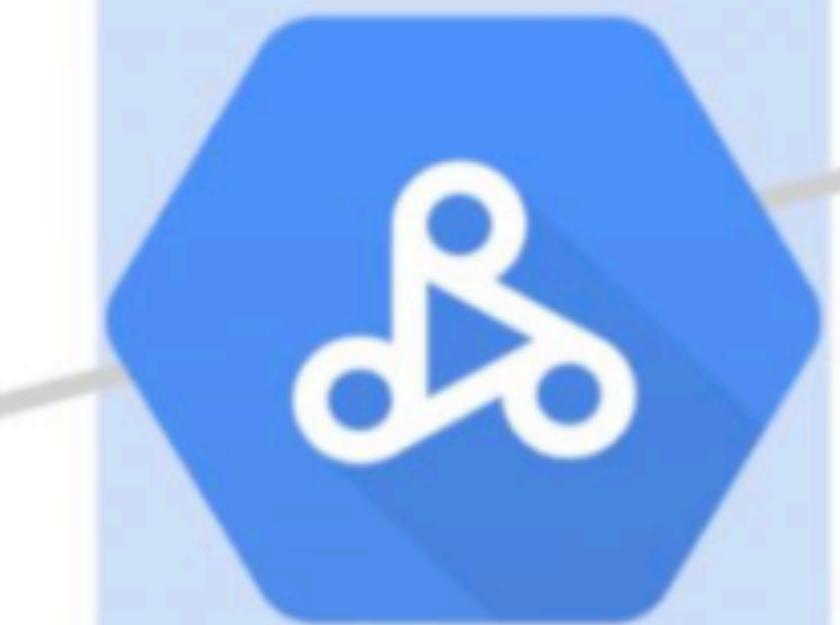
Low Rate &  
Billed By The  
Minute

**Less Waiting**

Clusters Start  
& Stop in  
< 90 Sec

**Minimal Ops**

We Manage  
Spark and  
Hadoop



**Google  
Cloud  
Dataproc**

Lower Cost &  
Complexity

**Higher Scale &  
Productivity**

**Easier  
To Use**

Manage  
Jobs With  
Easy Tools

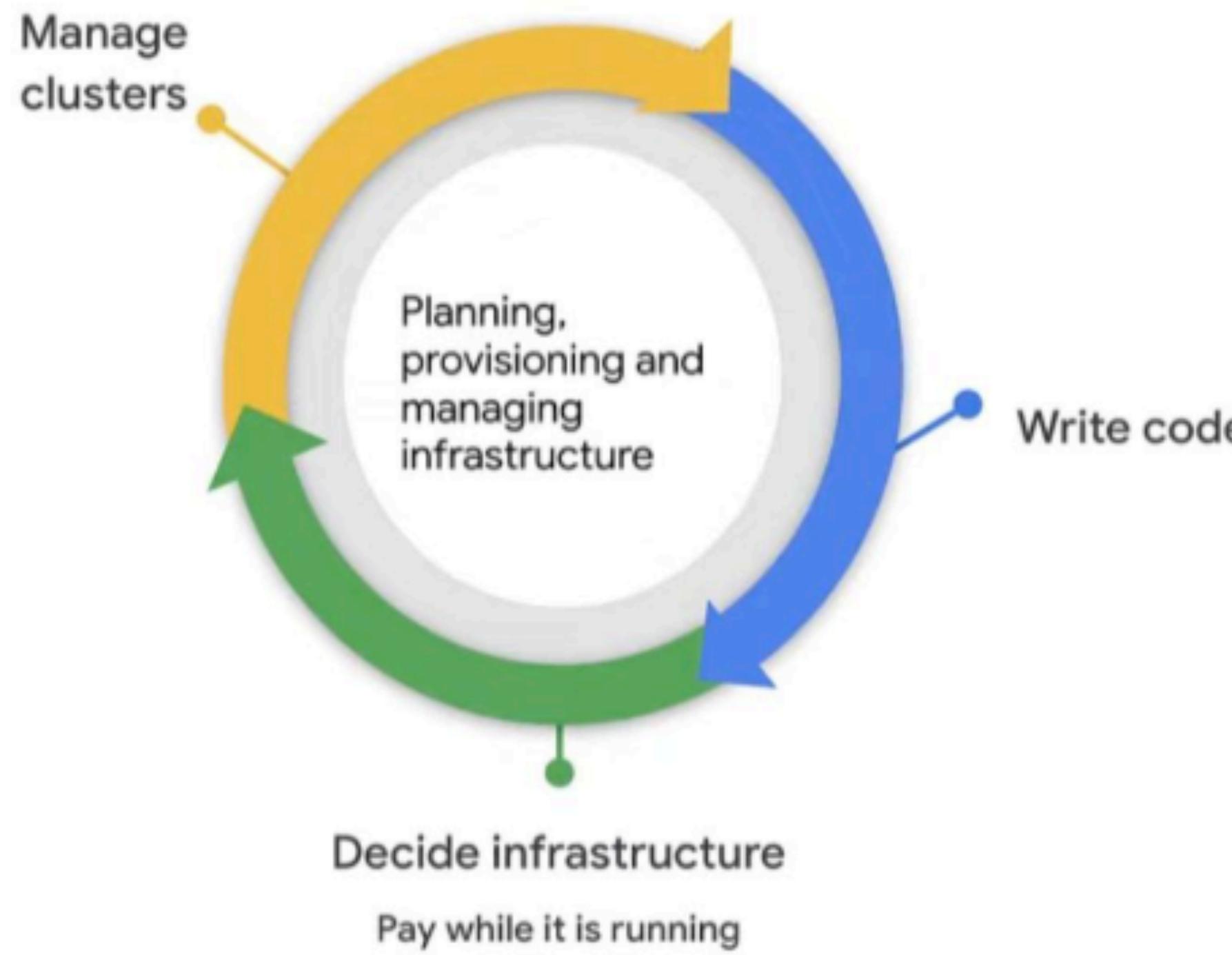
**Up To  
Date**

Recent Spark  
and Hadoop  
Releases

**Cloud  
Native**

Integrated w/  
Google Cloud  
Products

# What's better than Dataproc ? Serverless Spark!



- ✓ Job auto-scales
- ✓ No infrastructure to tune
- No clusters to manage
- Only pay for the job duration

**Focus on Spark, not infrastructure**

# **Dataproc serverless demo - Swipe**

# Dive deeper with codelabs



Head over to [g.co/codelabs/dataproc-serverless](https://g.co/codelabs/dataproc-serverless) to gain hands-on experience

Dataproc Serverless

⌚ 37 mins remaining | English | [English](#)

## Dataproc Serverless

**About this codelab**

Last updated Oct 12, 2022 | Written by Brad Miro

**1. Overview - Google Dataproc**

**Dataproc** is a fully managed and highly scalable service for running Apache Spark, Apache Flink, Presto, and many other open source tools and frameworks. Use Dataproc for data lake modernization, ETL / ELT, and secure data science, at planet scale. Dataproc is also fully integrated with several Google Cloud services including [BigQuery](#), [Cloud Storage](#), [Vertex AI](#), and [Dataplex](#).

Dataproc is available in three flavors:

- [Dataproc Serverless](#) allows you to run PySpark jobs without needing to configure infrastructure and autoscaling. Dataproc Serverless supports PySpark batch workloads and sessions / notebooks.
- [Dataproc on Google Compute Engine](#) allows you to manage a Hadoop YARN cluster for YARN-based Spark workloads in addition to open source tools such as Flink and Presto. You can tailor your cloud-based clusters with as much vertical or horizontal scaling as you'd like, including autoscaling.
- [Dataproc on Kubernetes Engine](#) allows you to configure Dataproc virtual clusters in your GKE infrastructure for submitting Spark, PySpark, SparkR or Spark SQL jobs.

Dataproc Serverless

37 minutes | Updated October 12, 2022 | [Start](#)

In this codelab, you'll learn all about Dataproc Serverless, including how to get started and how to access its rich featureset.

Dataproc on Google Compute Engine

16 minutes | Updated October 7, 2022 | [Start](#)

In this codelab, you will learn about using Dataproc on Google Compute Engine (GCE).

Create Spark ML models with Google Dataproc

31 minutes | Updated October 12, 2022 | [Start](#)

In this codelab, you'll submit Spark ML jobs to Google's Dataproc service.

Provisioning and Using a Managed Hadoop/Spark Cluster with Cloud Dataproc (Command Line)

20 minutes | Updated May 2, 2022 | [Start](#)

In this codelab, you will learn how to start a managed Spark/Hadoop cluster using Dataproc, submit a sample Spark job, and shut down your cluster using the command line.

PySpark for Natural Language Processing on Dataproc

42 minutes | Updated January 24, 2022 | [Start](#)

This lab shows you how to use PySpark on Dataproc to load data from BigQuery and save it to Google Cloud Storage.

Apache Spark and Jupyter Notebooks on Cloud Dataproc

32 minutes | Updated June 25, 2021 | [Start](#)

This lab shows you how to use Spark MLlib and spark-nlp for performing machine learning and NLP on large quantities of data.

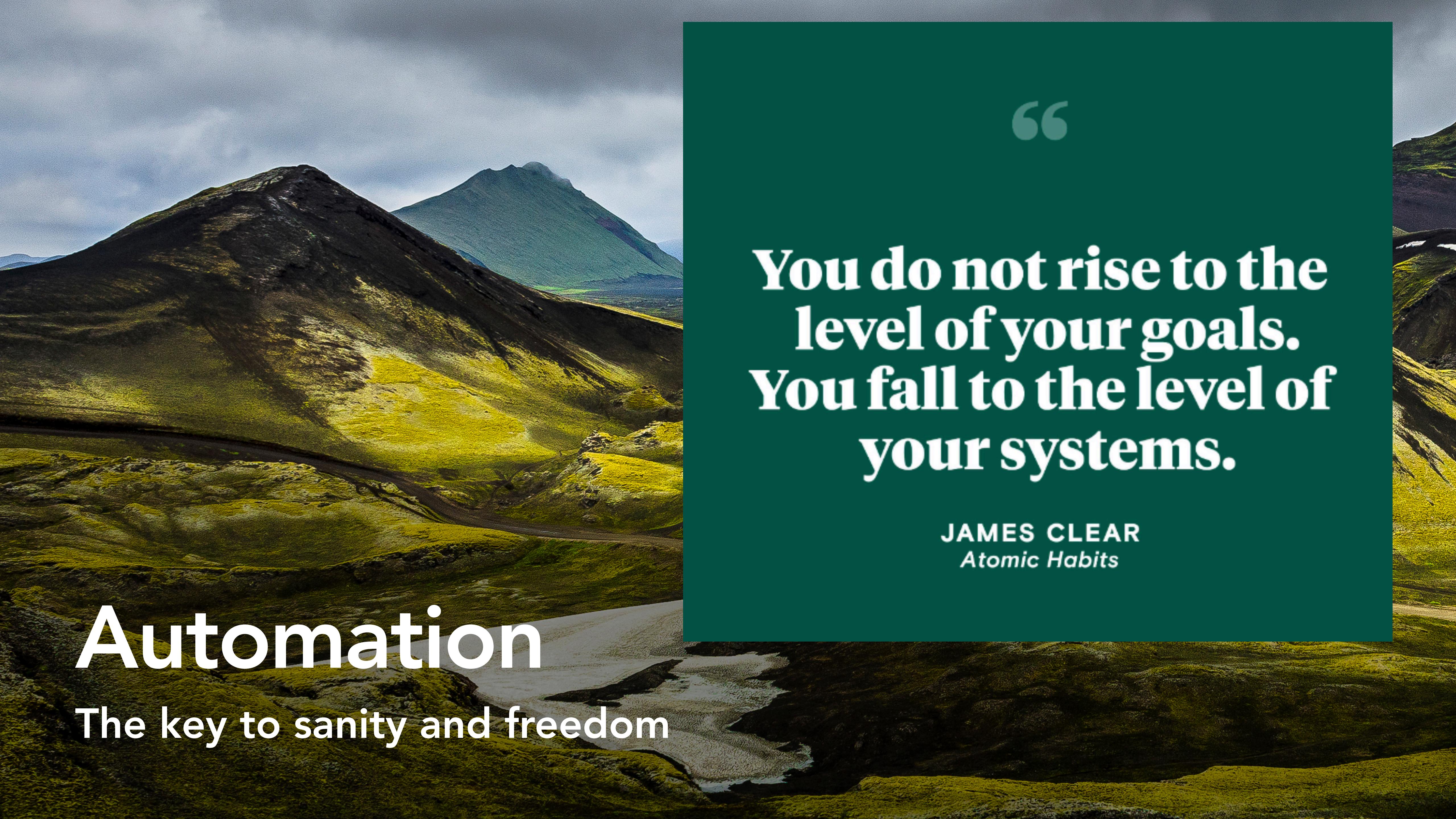
AI Speech Recognition with TensorFlow Lite for Microcontrollers and SparkFun Edge

1 hour | Updated October 13, 2020 | [Start](#)

# We needed

Break down business problems into actionable items

- Auto scaling clusters/nodes + Cost effective (Small budget team)
  - Dataproc and Dataproc Serverless
- Automation of our cron jobs
- ??



# Automation

## The key to sanity and freedom

“

You do not rise to the  
level of your goals.  
You fall to the level of  
your systems.

JAMES CLEAR  
*Atomic Habits*

# Enter Airflow

Airflow    DAGs    Security    Browse    Admin    Docs    21:11 UTC    RH

## DAGs

All 26	Active 10	Paused 16	Filter DAGs by tag	Search DAGs			
DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links
example_bash_operator	airflow	2	0 0 * * *	2020-10-26, 21:08:11	5	▶ C ⚡	...
example_branch_dop_operator_v3	airflow	0	@/1 * * * *		0	▶ C ⚡	...
example_branch_operator	airflow	1	@daily	2020-10-23, 14:09:17	11	▶ C ⚡	...
example_complex	airflow	1	None	2020-10-26, 21:08:04	37	▶ C ⚡	...
example_external_task_marker_child	airflow	1	None	2020-10-26, 21:07:33	2	▶ C ⚡	...
example_external_task_marker_parent	airflow	1	None	2020-10-26, 21:08:34	1	▶ C ⚡	...
example_kubernetes_executor	airflow	0	None		0	▶ C ⚡	...
example_kubernetes_executor_config	airflow	1	None	2020-10-26, 21:07:40	5	▶ C ⚡	...
example_nested_branch_dag	airflow	1	@daily	2020-10-26, 21:07:37	9	▶ C ⚡	...
example_passing_params_via_test_command	airflow	0	@/1 * * * *		0	▶ C ⚡	...

I wish I knew of this in 2016

# DAGs

## What are they ?

what we will look for in the airflow DAG list

```
with DAG(  
    "my_dag_name", start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),  
    schedule_interval="@daily", catchup=False  
) as dag:  
    op = EmptyOperator(task_id="task")
```

Any operator can be used

# DAGs

```
"""Example DAG demonstrating the usage of the BashOperator."""

import datetime
from airflow import DAG
from airflow.operators.bash import BashOperator

with DAG(
    dag_id='GCS_BQ_bash_operator',
    schedule_interval='0 0 * * *',
    start_date=datetime.datetime(2022, 8, 18),
    dagrun_timeout=datetime.timedelta(minutes=60),
    tags=['gcs', 'bq'],
) as dag:
    run_this_1 = BashOperator(
        task_id='run_first',
        bash_command='bq mk mydataset',
    )
    run_this_2 = BashOperator(
        task_id='run_after',
        bash_command='bq load --autodetect --source_format=CSV mydataset.mytable gs://bucket_name/table_name.xls',
    )

    run_this_3 = BashOperator(
        task_id='run_last_for_cleanup',
        bash_command='bq rm airflowTPCH',
    )

    run_this_1 >> run_this_2 >> run_this_3

if __name__ == "__main__":
    dag.cli()
```

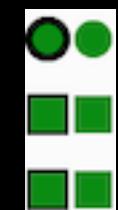
# Tasks

## Happy flow

1. No status (scheduler created empty task instance)
2. Scheduled (scheduler determined task instance needs to run)
3. Queued (scheduler sent task to executor to run on the queue)
4. Running (worker picked up a task and is now running it)
5. Success (task completed)

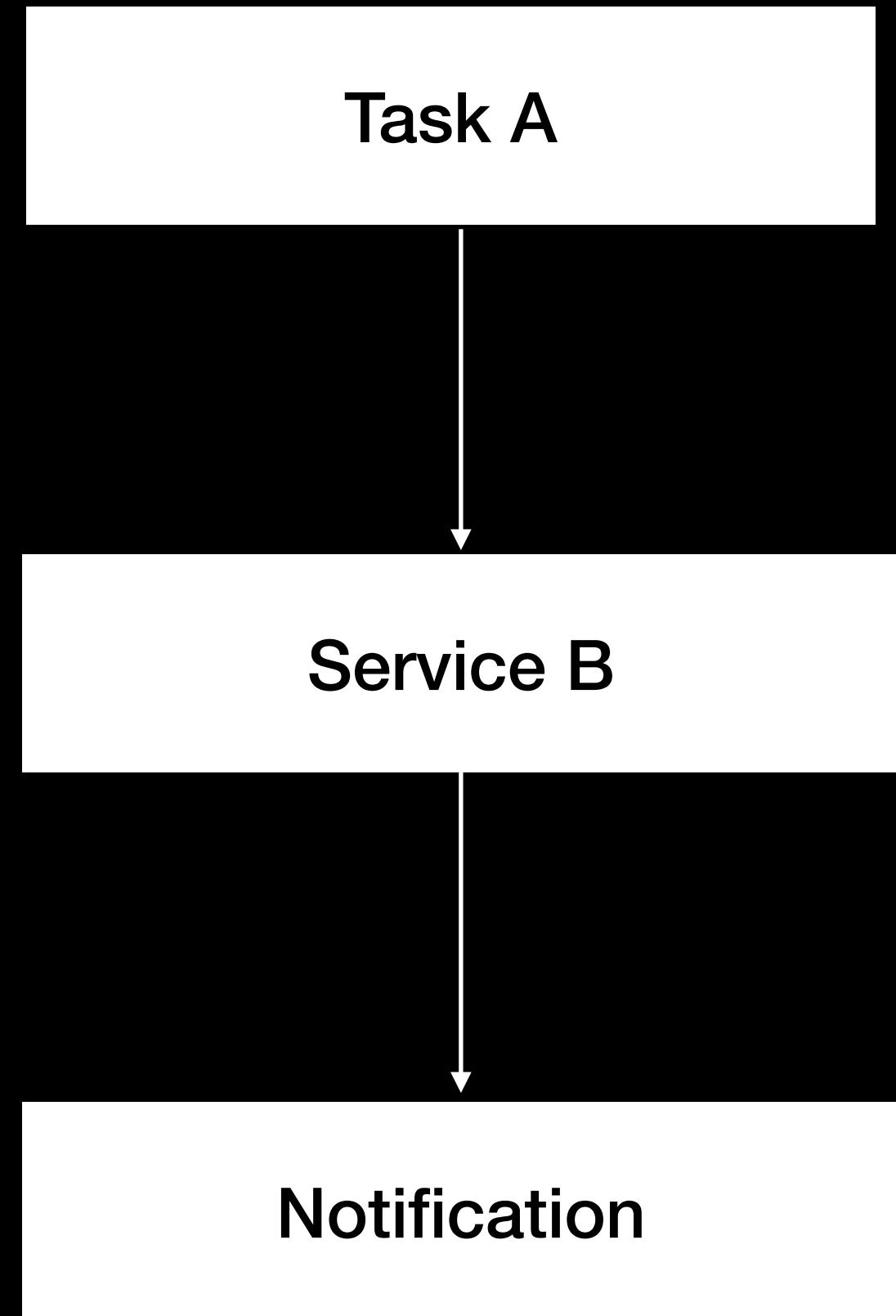
```
with DAG('my_dag', start_date=datetime(2016, 1, 1)) as dag:  
    task_1 = DummyOperator('task_1')  
    task_2 = DummyOperator('task_2')  
    task_1 >> task_2 # Define dependencies
```

 success  running  failed  skipped  rescheduled  retry  queued  no status



# Composer

- Why
- Under the hood
- Dataproc Operators demo



# Why Composer



Did someone say multi-cloud?

- Managed Airflow
- Focus on your workflows and not your infrastructure
- Since its based on open-source framework your workflows can be multi-cloud and/or on-prem
- Uses Python - so no new lang

# Under the hood

- Airflow workers are executed on GKE clusters
- Airflow metadata is stored on Cloud SQL
- Airflow Webserver runs as an App Engine Flex which is protected by an Identity Aware Proxy



Composer		Environments		+ CREATE	- DELETE							
Filter environments												
	Name	Location	Composer version	Airflow version	Creation time	Update time	Airflow webserver	DAG list	NEW	Logs	DAGs folder	Labels
<input type="checkbox"/>	<a href="#">testing-bq</a>	us-central1	2.0.23	2.2.5	8/18/22, 12:12 PM	8/18/22, 12:12 PM	None	<a href="#">DAGs</a>	<a href="#">Logs</a>	None	None	None



Airflow webserver	DAG list	Logs	DAGs folder	Labels
<a href="#">Airflow</a>	<a href="#">DAGs</a>	<a href="#">Logs</a>	<a href="#">DAGs</a>	None

[Airflow](#)

[DAGs](#)

[Logs](#)

[DAGs](#)

None



Go here to see the airflow dashboard



Add your DAG python file in this folder

**Dataproc - Composer demo - Swipe**

# Recap

## Solutions we found for our problems

- Auto scaling clusters/nodes + Cost effective (Small budget team)
  - Dataproc
- Automation of our cron jobs
  - Composer

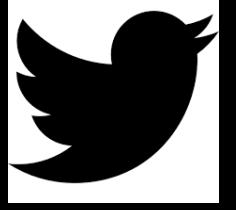
# Further reading

- [jsj14.medium.com](https://jsj14.medium.com)
- Composer codelabs
- Documentation

# Thank you

Please take 3 seconds to fill a feedback form for me - [bit.ly/julian-feedback](https://bit.ly/julian-feedback)

- [linkedin.com/in/julian-s-joseph/](https://linkedin.com/in/julian-s-joseph/)

- justjulian14 
- juliansarajoseph 

**Be smarter and less scared of complex things.**