Mineração

Projeto Final de Engenharia de Dados

Roteiro

01 Apresentando o setor

02 Conhecendo os dados

03 Requisitos do projeto

04 Stack

05 Análises

06 Pipeline

07 Desenvolvimento

08 Avaliação

Time de Projeto

Diego Alves

Porto Alegre - Rio Grande do Sul

Rafael Cicaroli

São Paulo

Rayssa Alcântara

João Pessoa - Paraíba

Sandi Ourique

Ijuí - Rio Grande do Sul

Apresentando o setor

Conceito

Mineração é a extração de minerais valiosos ou outros materiais geológicos da Terra, geralmente de um local onde o minério é encontrado.

Apresentando o setor

Importância

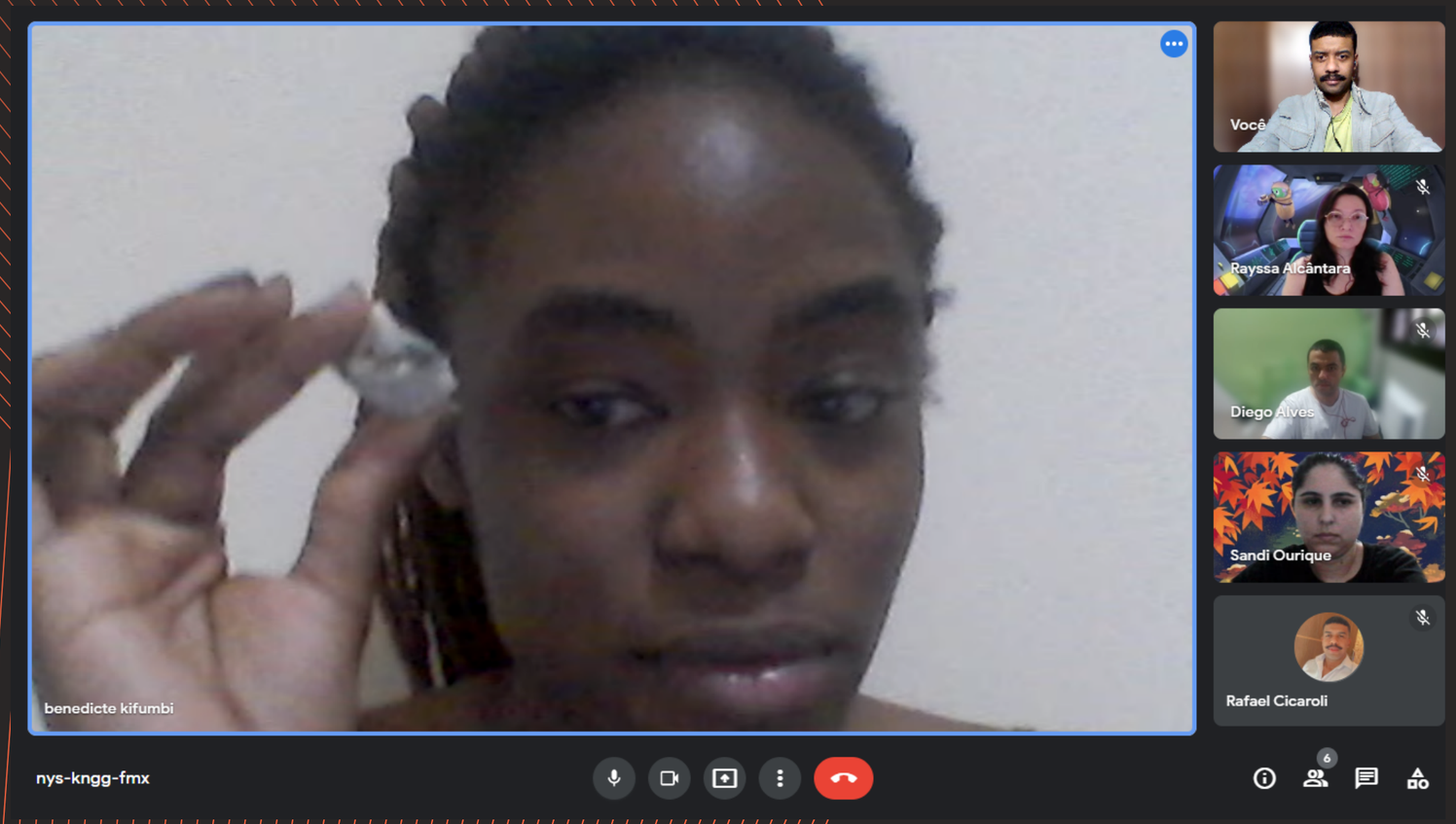
Além da geração de empregos, arrecadação de impostos e exportação de produtos, os minérios estão presentes em praticamente qualquer atividade da vida moderna.

Apresentando o setor

Compreensão

- Pesquisa
- Bate Papo Geóloga
- Consolidação

Levantamento de Requisitos



Reunião com Geóloga

- Termos Técnicos
- Referências

Conhecendo os dados

Conhecer e entender os dados disponíveis, avaliação de sua qualidade e compatibilidade entre volume e requisitos do negócio.

ANM 2.418.560

arrecadacao (1.657.295), autuacao (20.539), barragens (907), beneficiada (2.684), distribuicao (739.819)

IBGE 100.189

municipio (5.572), pib (94.617)

TOTAL

2.518.749

Requisitos do Projeto

Desenvolvimento da solução com computação em nuvem

Uso de banco de dados

Processamento em Cluster

Stack

PostgreSQL

cassandra

APACHE Spark

python

Google Cloud

Análises

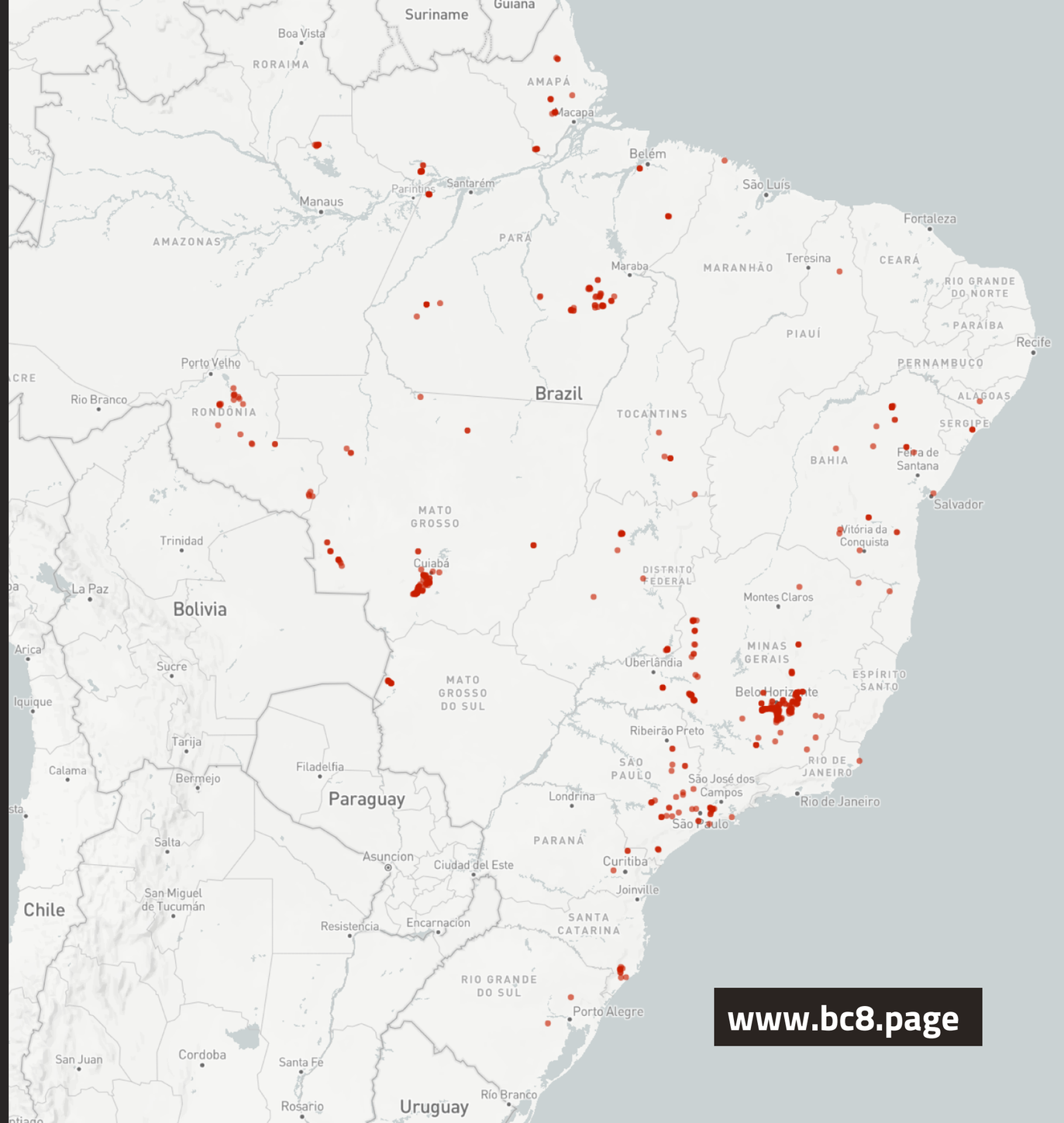
Barragens

no Brasil

275 Mineradoras

907 Barragens

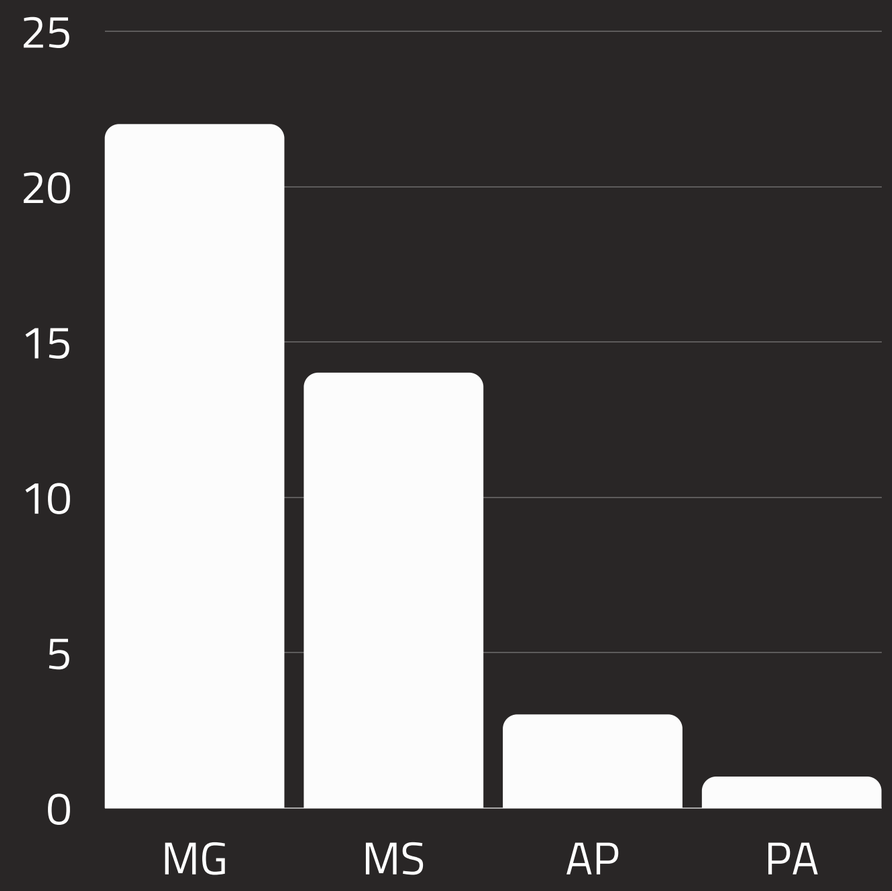
205 Minérios



www.bc8.page

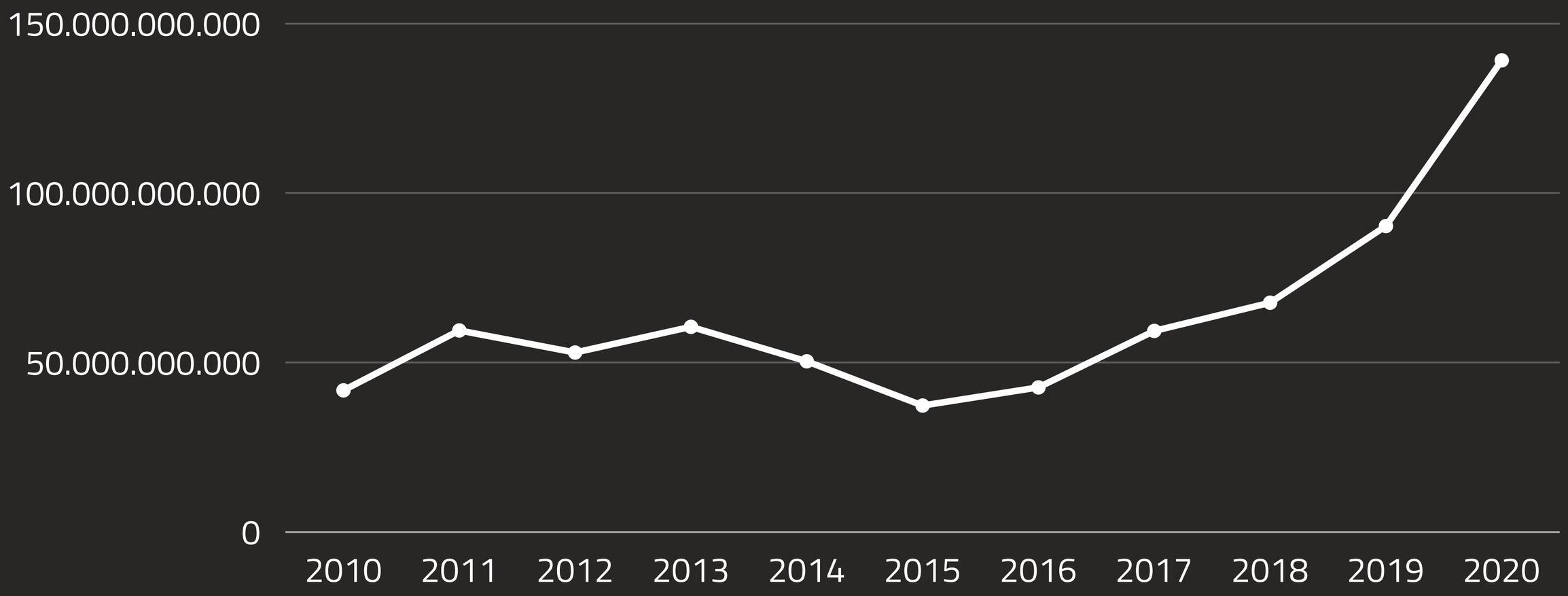
Barragens ativas

Mineração de Ferro



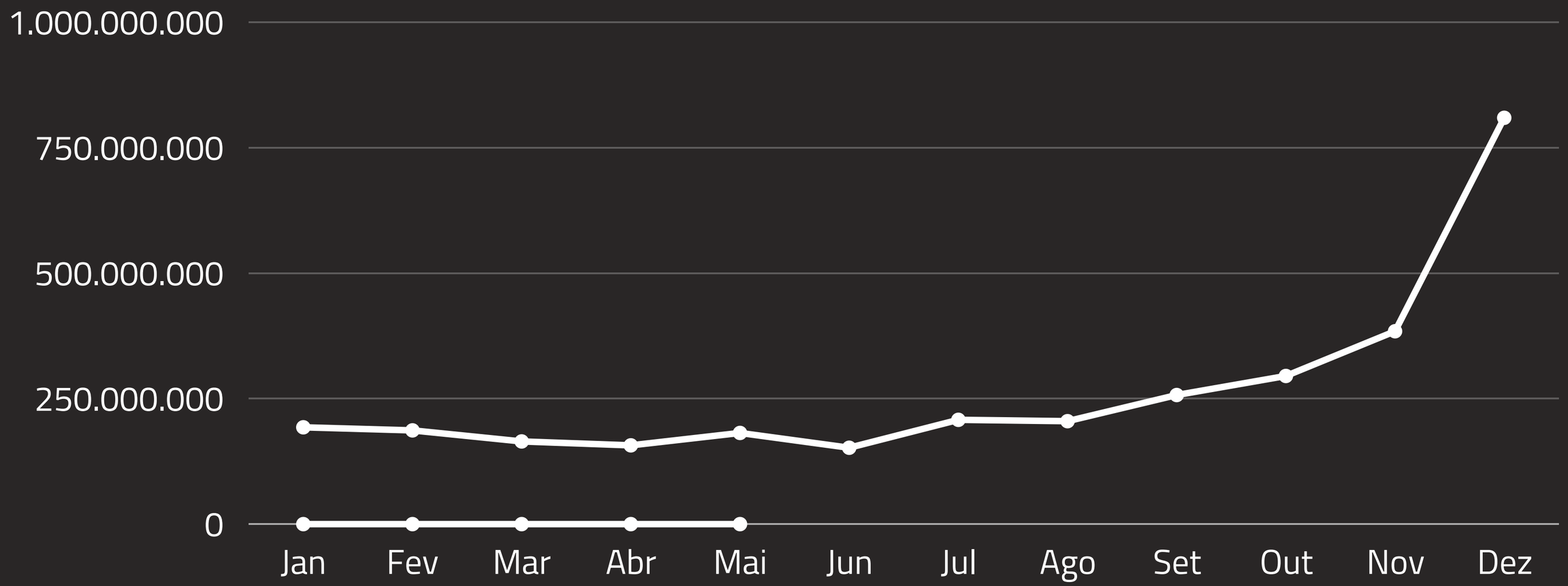
Venda do Ferro

entre 2010 e 2020 em Reais



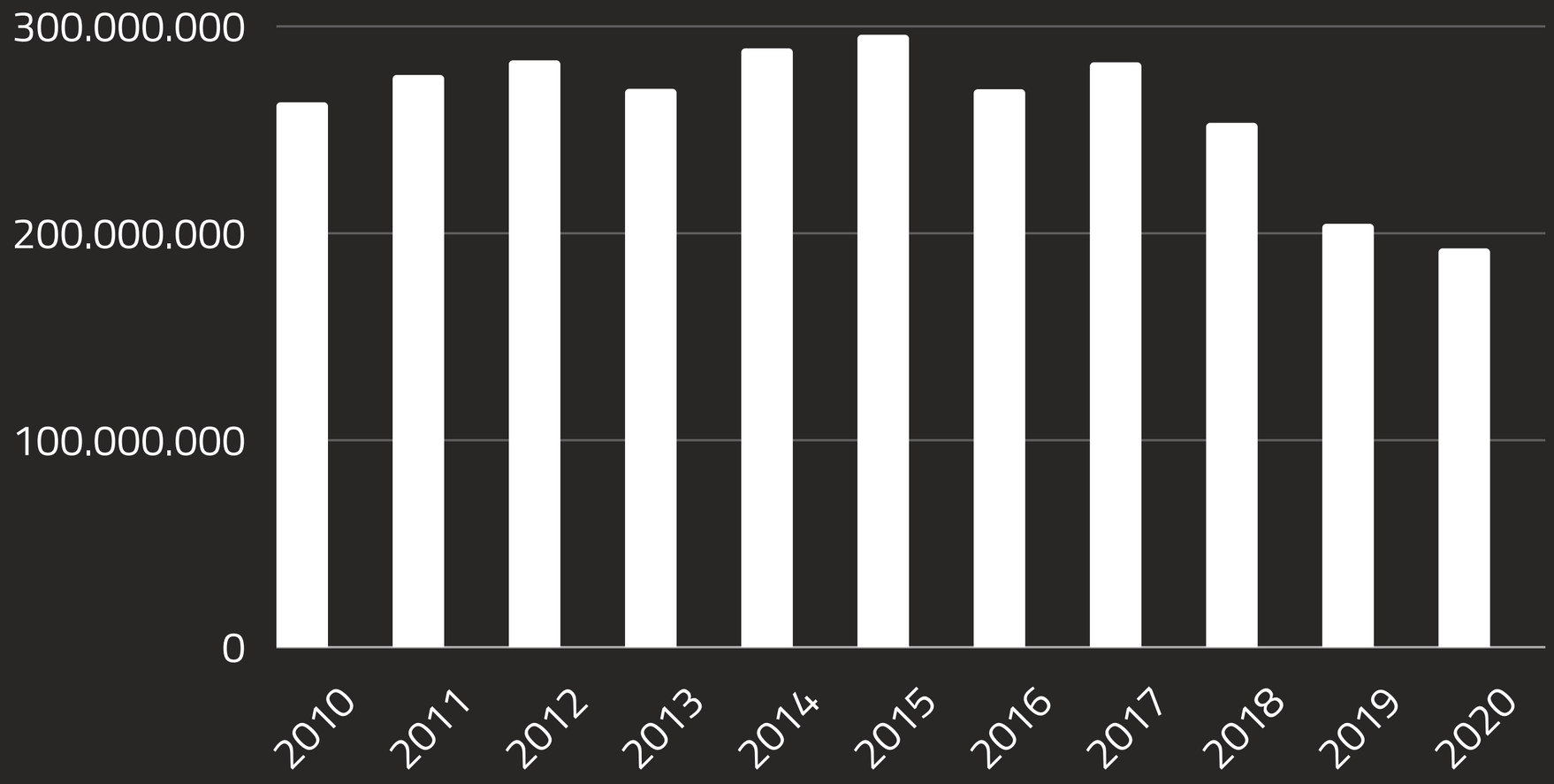
Produção Ferro

no ano de 2020 em tolenadas



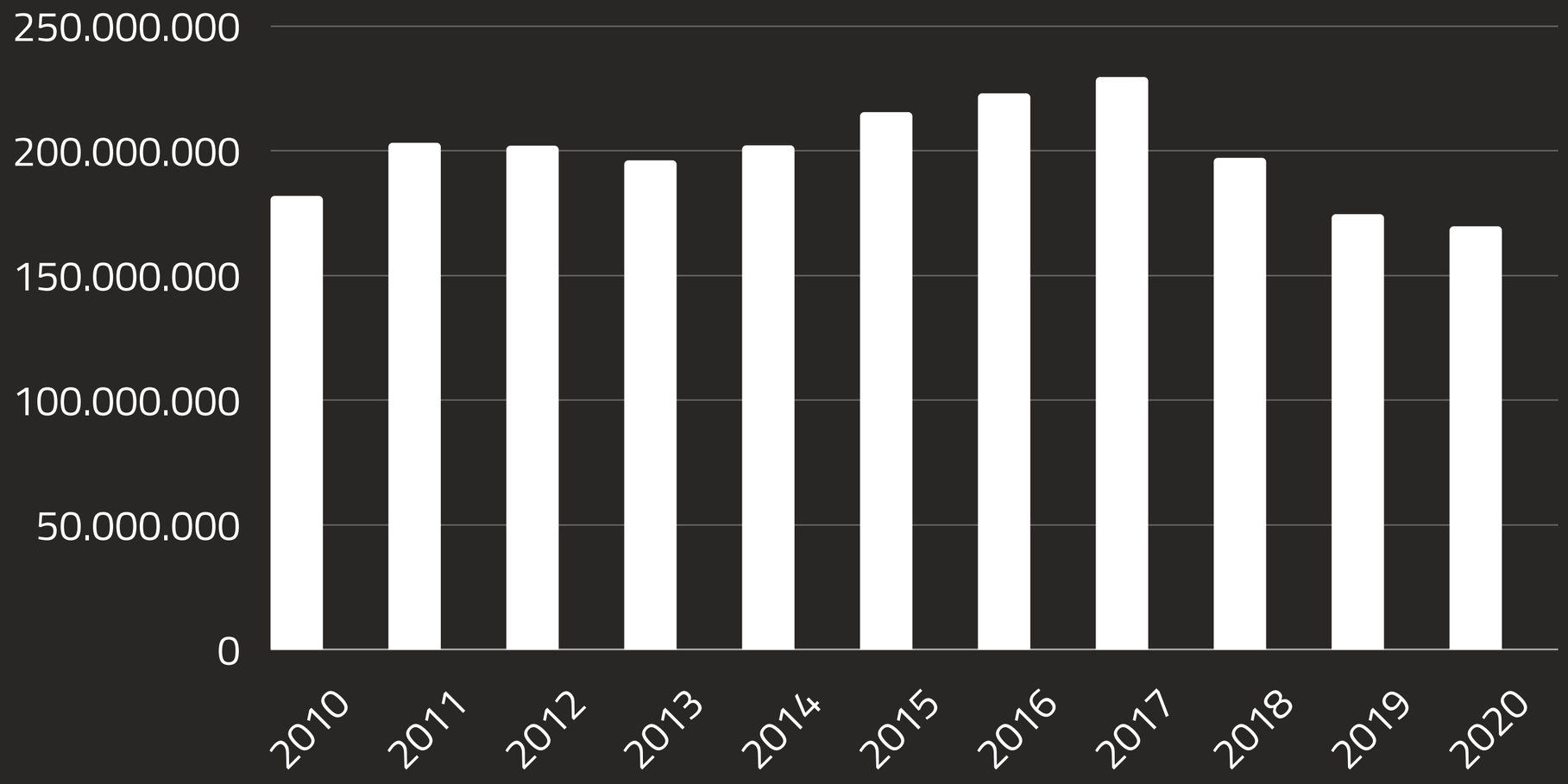
Produção ferro

Minas Gerais entre 2010 - 2020 em toneladas



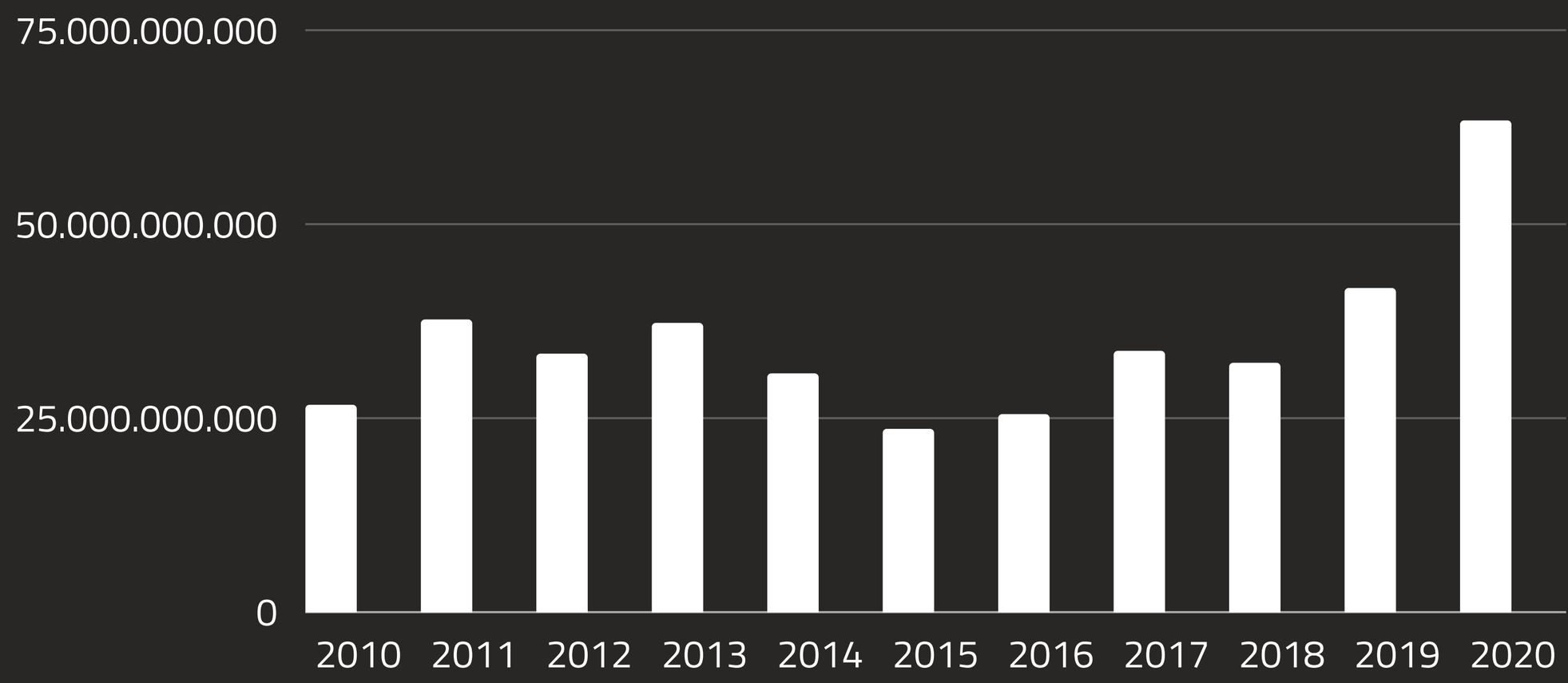
Volume de vendas

Minas Gerais entre 2010 - 2020 em Toneladas



Total Comercializado

Minas Gerais entre 2010 - 2020 em R$



Carvão

Brasil **0,1%**

Desde **1827**

? **2022**

Impacto

das Mineradoras que produzem Carvão

- Copelmi Mineração Ltda
- Indústria Carbonífera Rio Deserto Ltda
- Carbonífera Siderópolis Ltda
- Gabriella Mineração Ltda
- Carbonífera Belluno Ltda
- Companhia Riograndense de Mineração

Classe II A

Impacto

das Mineradoras que produzem Carvão

Copelmi Mineração Ltda

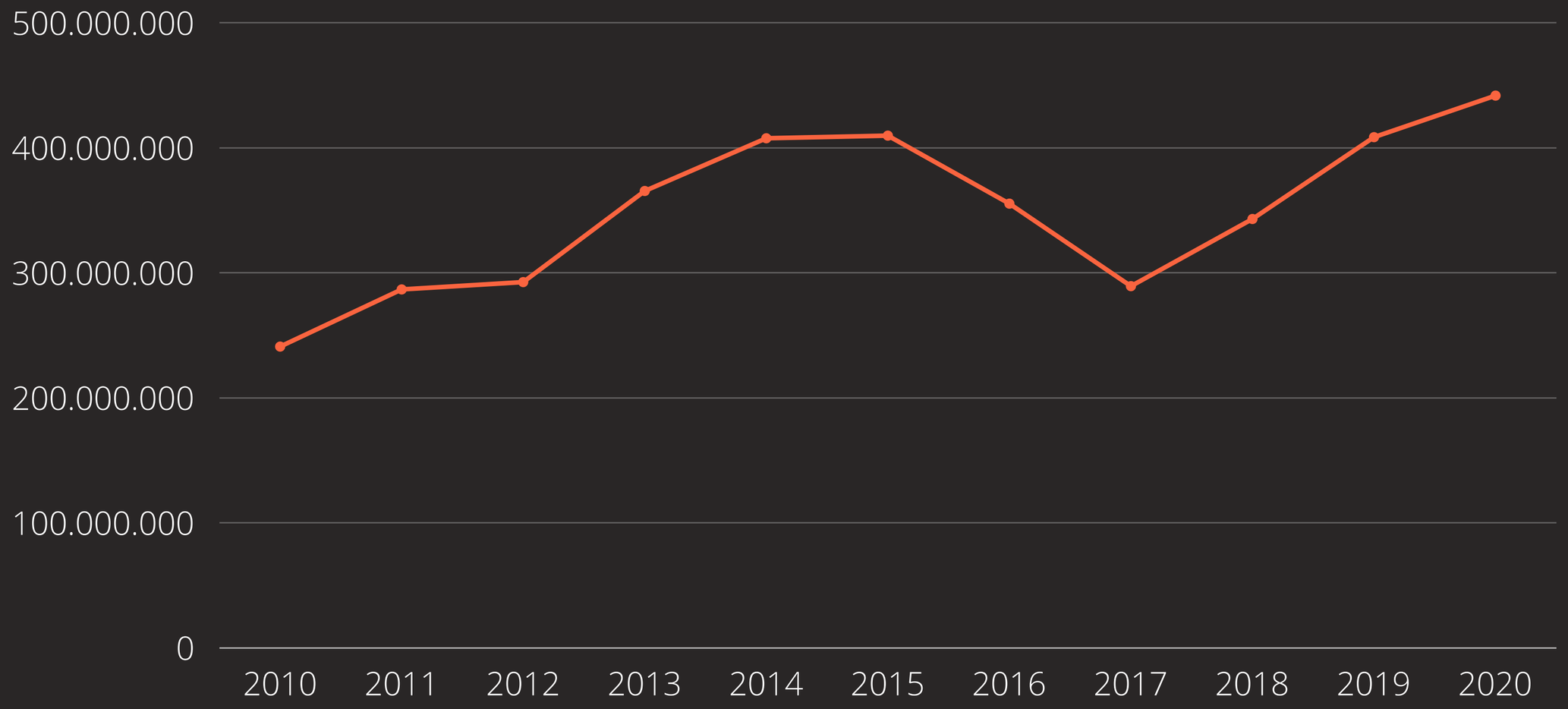
80% uso industrial

18% total do mercado

Responsabilidade Socioambiental

Carvão

Preço do Carvão no Estado de Maior produção do Brasil entre 2010 - 2020



Impacto

da Mineração de Ferro da Vale

- Forquilha I
- Campo Grande
- Timbopeba
- Xingu
- Peneirinha
- Norte/Laranjeiras
- Galego
- Forquilha IV

Existente

Existem pessoas ocupando permanentemente a área afetada a jusante da barragem, portanto, vidas humanas poderão ser atingidas

Município

da Barragem Forquilha I

Ouro Preto

74.824

Usiminas

Situação Operacional Barragens de Minério de Ferro

Dique Flotação

Em Operação

Barragem Central 06/06/2014

Desativada

Barragem Samambaia 26/12/2021

Desativada

Usiminas

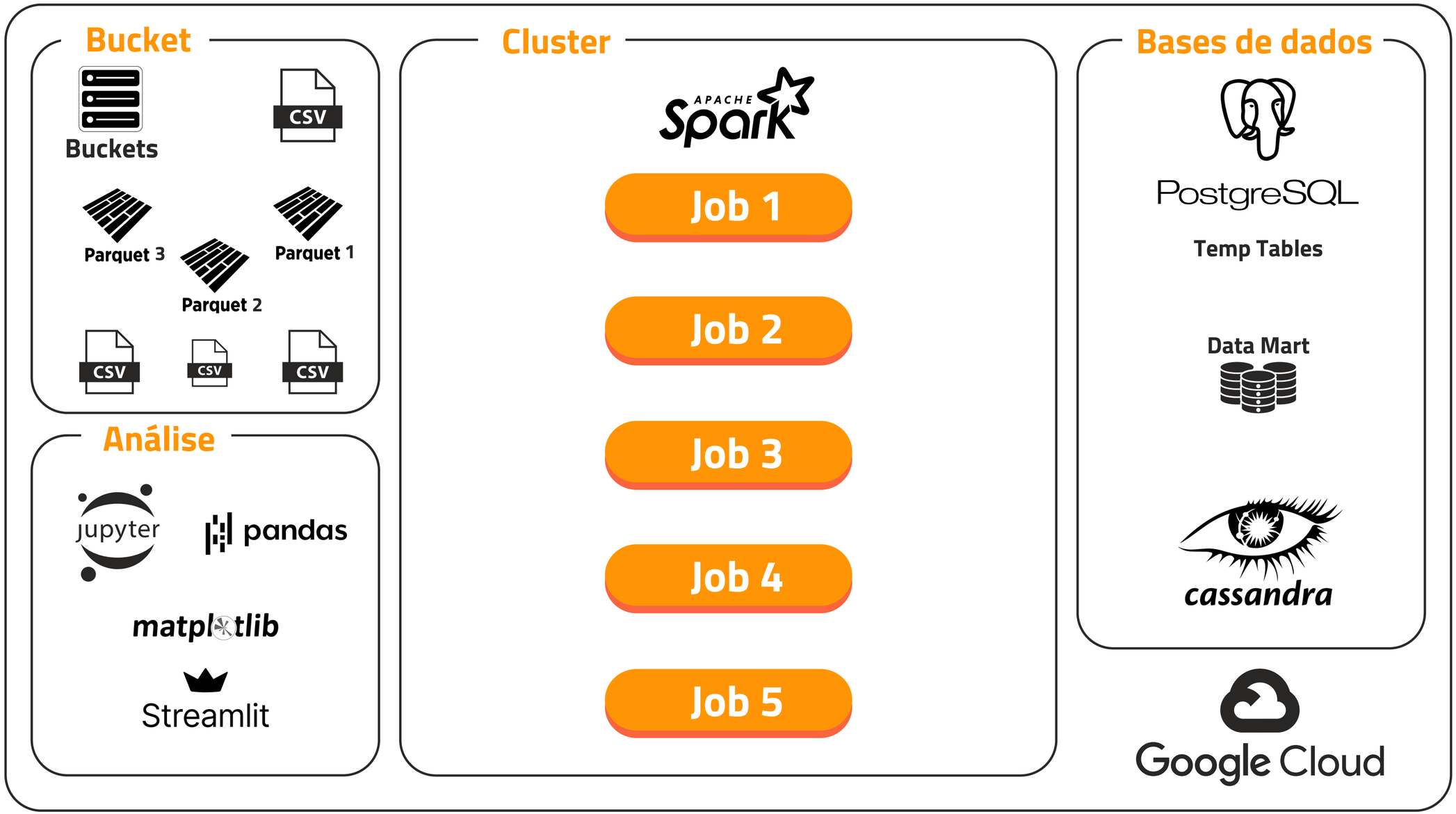
Situação Operacional Barragens de Minério de Ferro

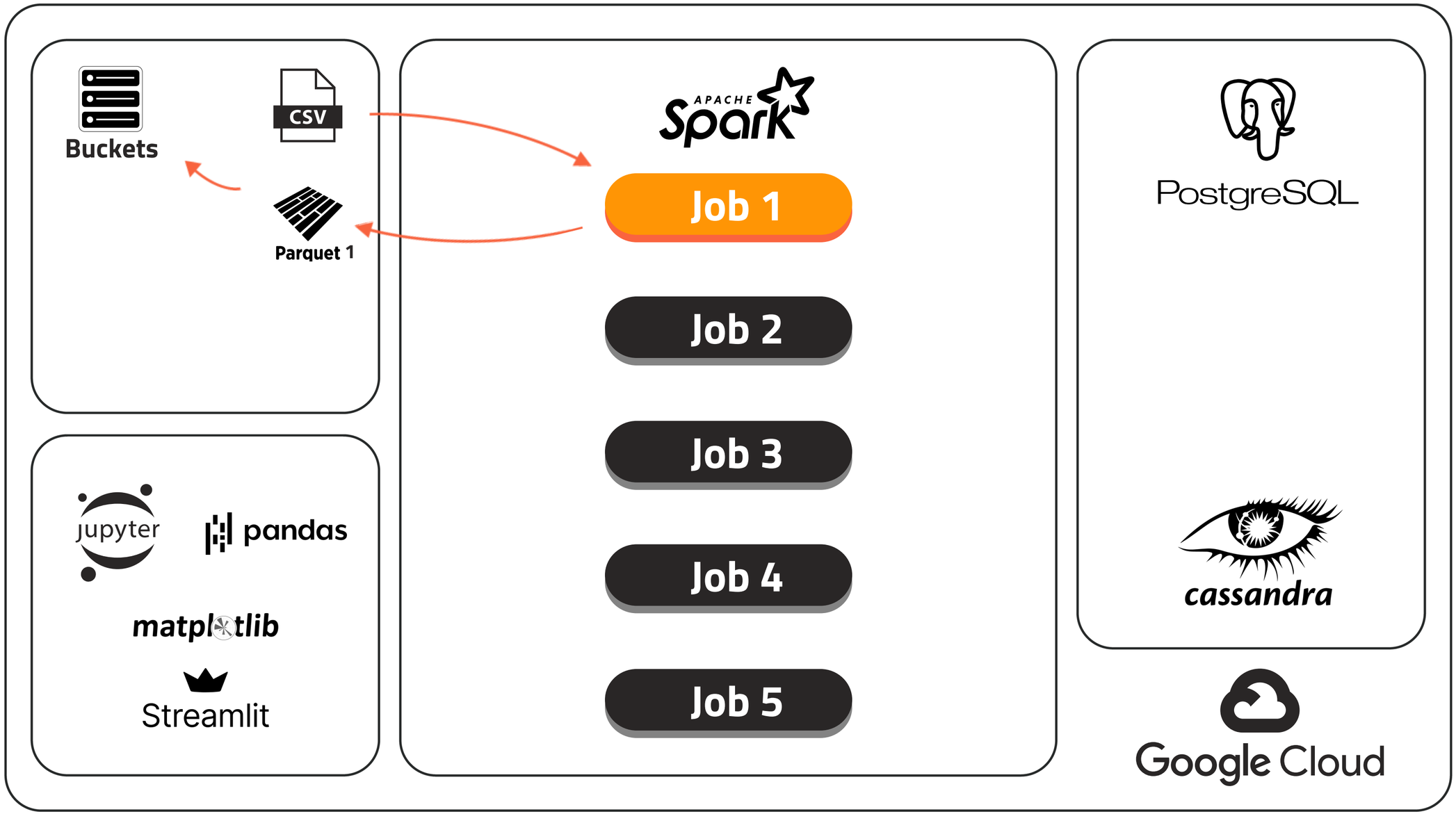
Barragem Samambaia **26/12/2021**

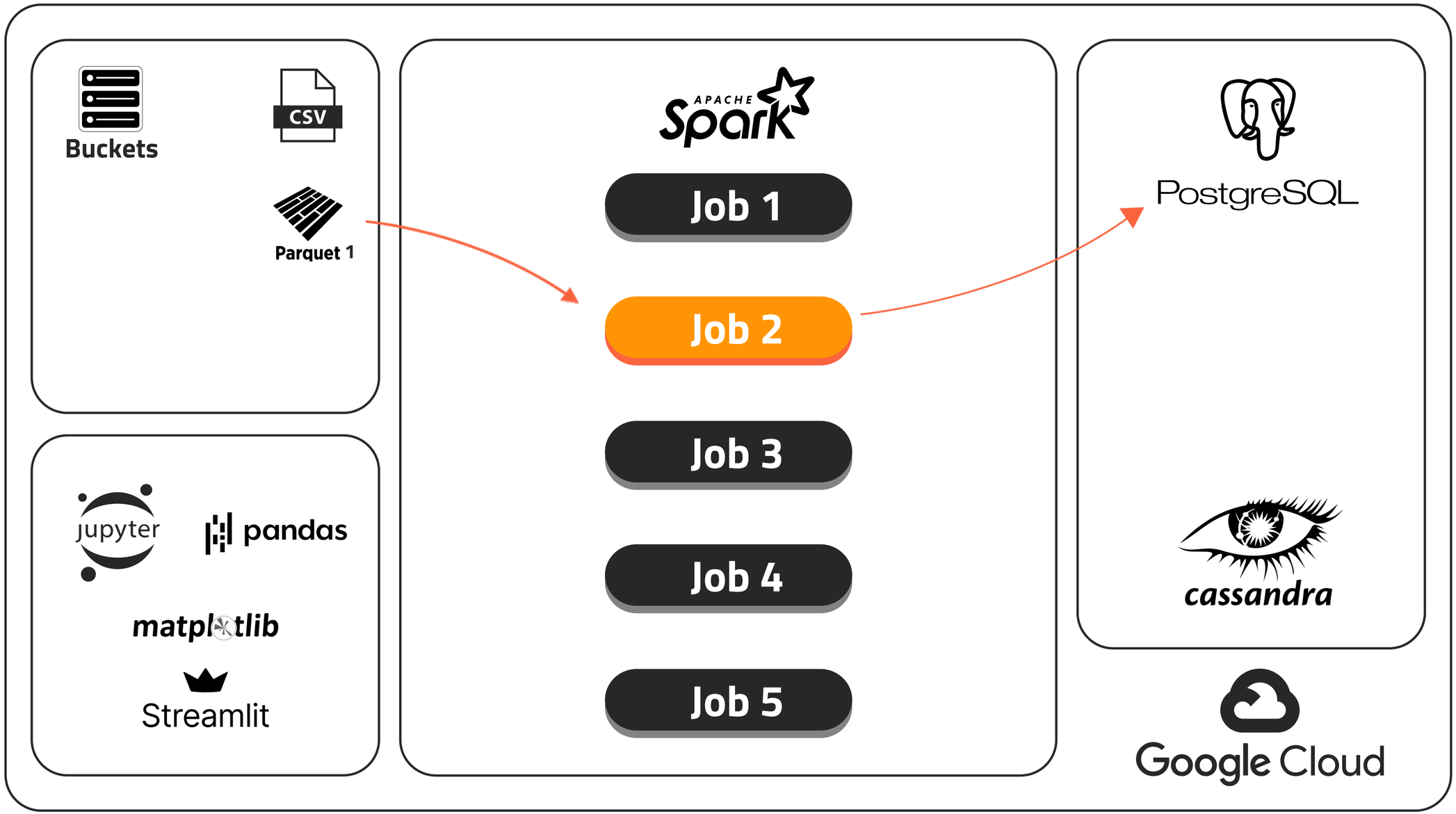
Desativada

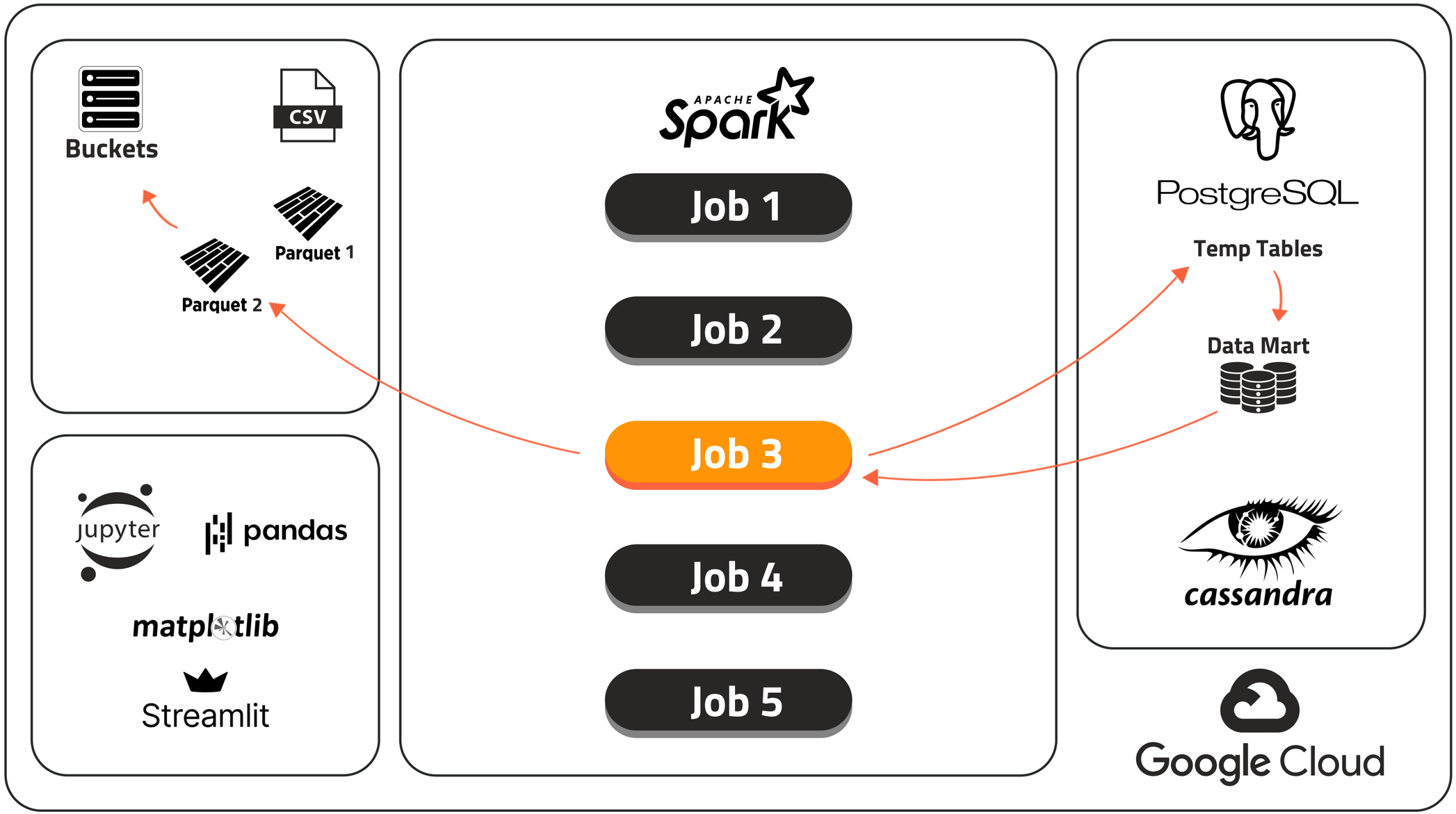
643 Barragens

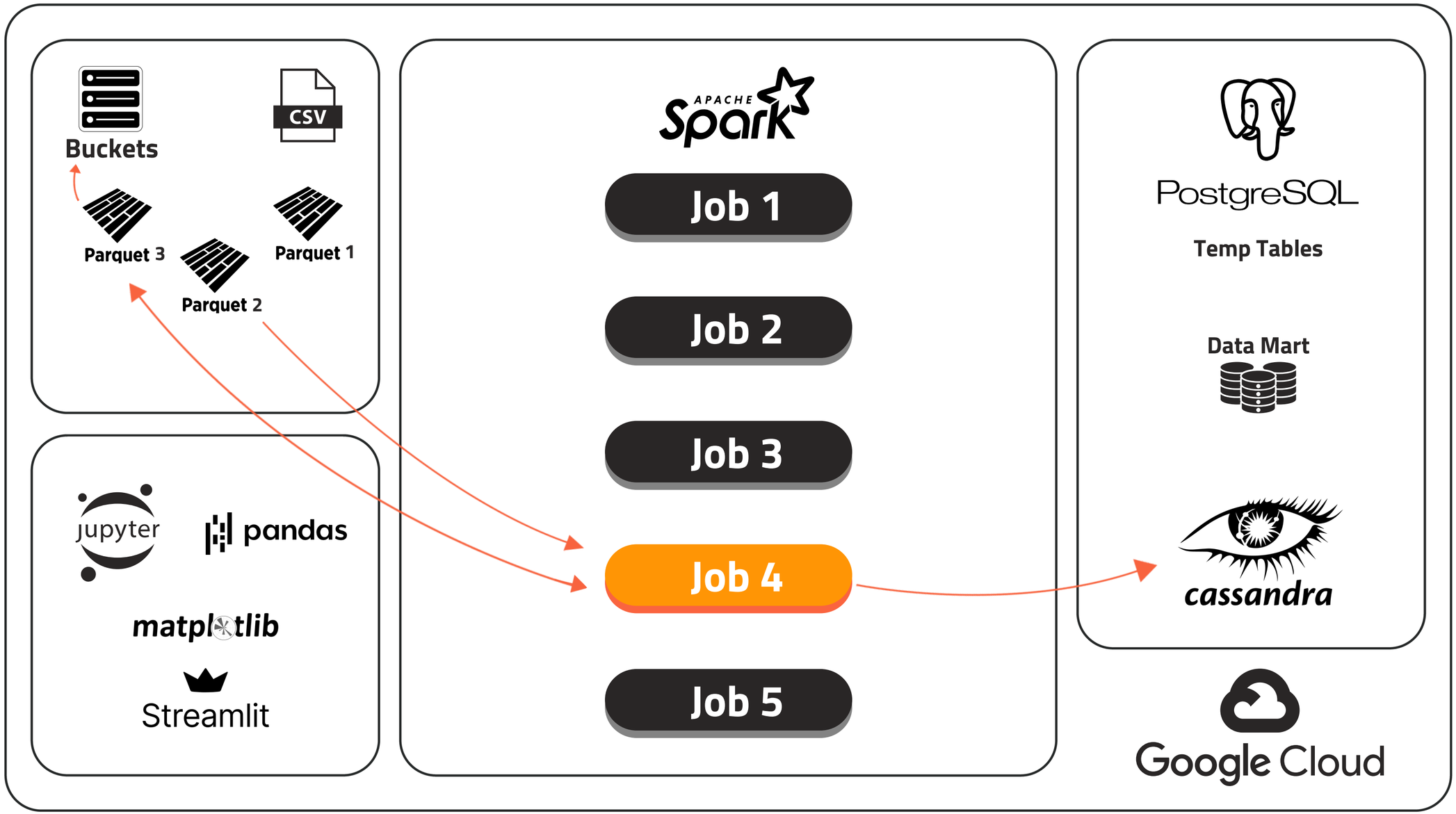
em plena operação no Brasil

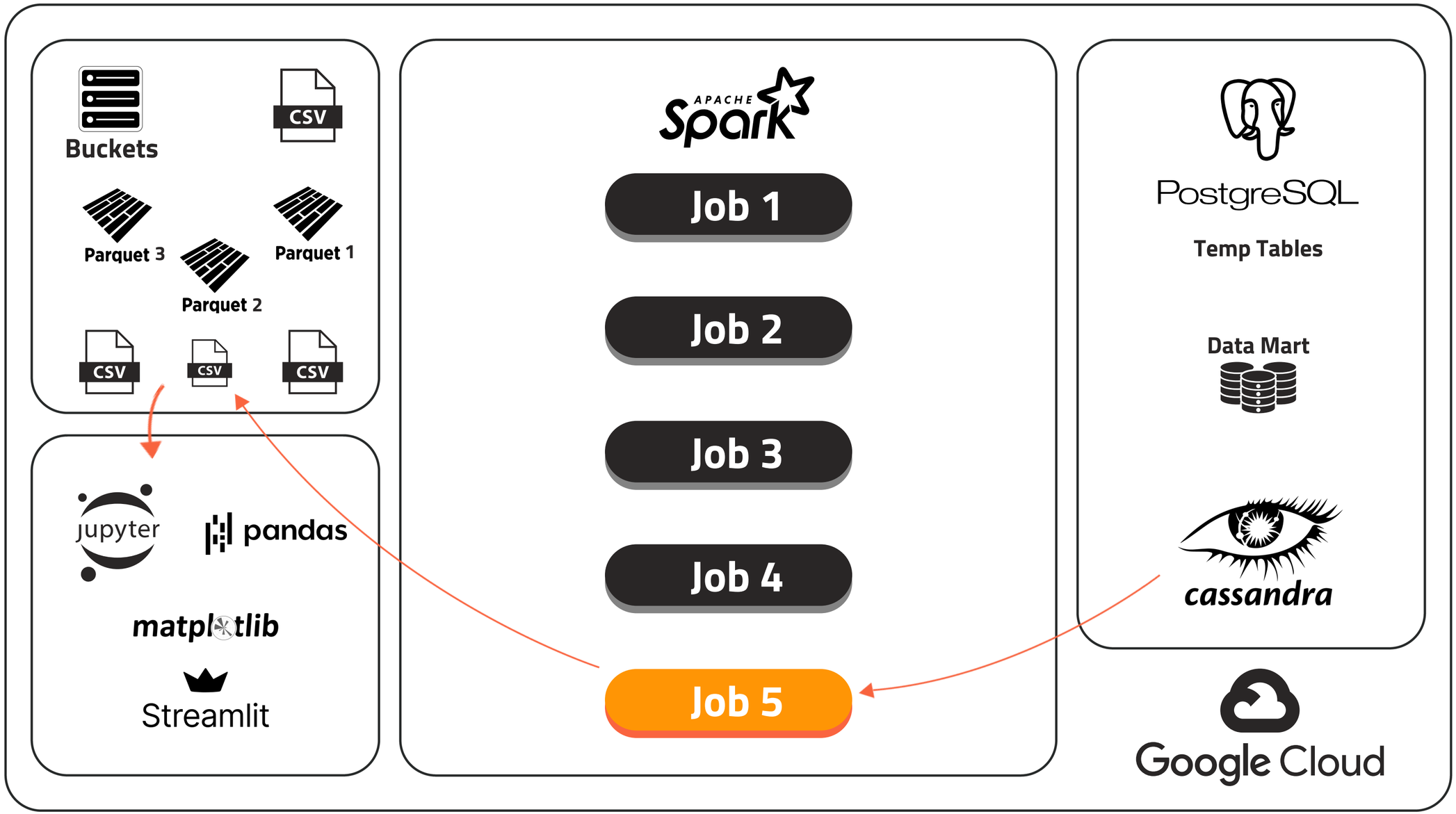

Bucket
Buckets
CSV
Parquet 3
Parquet 1
Parquet 2
CSV
CSV
CSV
Análise
jupyter
pandas
matplotlib
Streamlit
Cluster
APACHE Spark
Job 1
Job 2
Job 3
Job 4
Job 5
Bases de dados
PostgreSQL
Temp Tables
Data Mart
cassandra
Google Cloud


Buckets
CSV
Parquet 1
APACHE Spark
Job 1
Job 2
Job 3
Job 4
Job 5
PostgreSQL
cassandra
Google Cloud
jupyter
pandas
matplotlib
Streamlit


Buckets
CSV
Parquet 1
APACHE Spark
Job 1
Job 2
Job 3
Job 4
Job 5
PostgreSQL
jupyter
pandas
matplotlib
Streamlit
cassandra
Google Cloud


Buckets
CSV
Parquet 1
Parquet 2
jupyter
pandas
matplotlib
Streamlit
APACHE Spark
Job 1
Job 2
Job 3
Job 4
Job 5
PostgreSQL
Temp Tables
Data Mart
cassandra
Google Cloud


Buckets
CSV
Parquet 3
Parquet 2
Parquet 1
jupyter
pandas
matplotlib
Streamlit
APACHE Spark
Job 1
Job 2
Job 3
Job 4
Job 5
PostgreSQL
Temp Tables
Data Mart
cassandra
Google Cloud


Buckets
CSV
Parquet 3
Parquet 1
Parquet 2
CSV
CSV
CSV
jupyter
pandas
matplotlib
Streamlit
APACHE Spark
Job 1
Job 2
Job 3
Job 4
Job 5
PostgreSQL
Temp Tables
Data Mart
cassandra
Google Cloud

Diagrama

Caso de Uso

Ler CSV

Gravar Parquet

Ler Parquet

Inserir dados Postgres

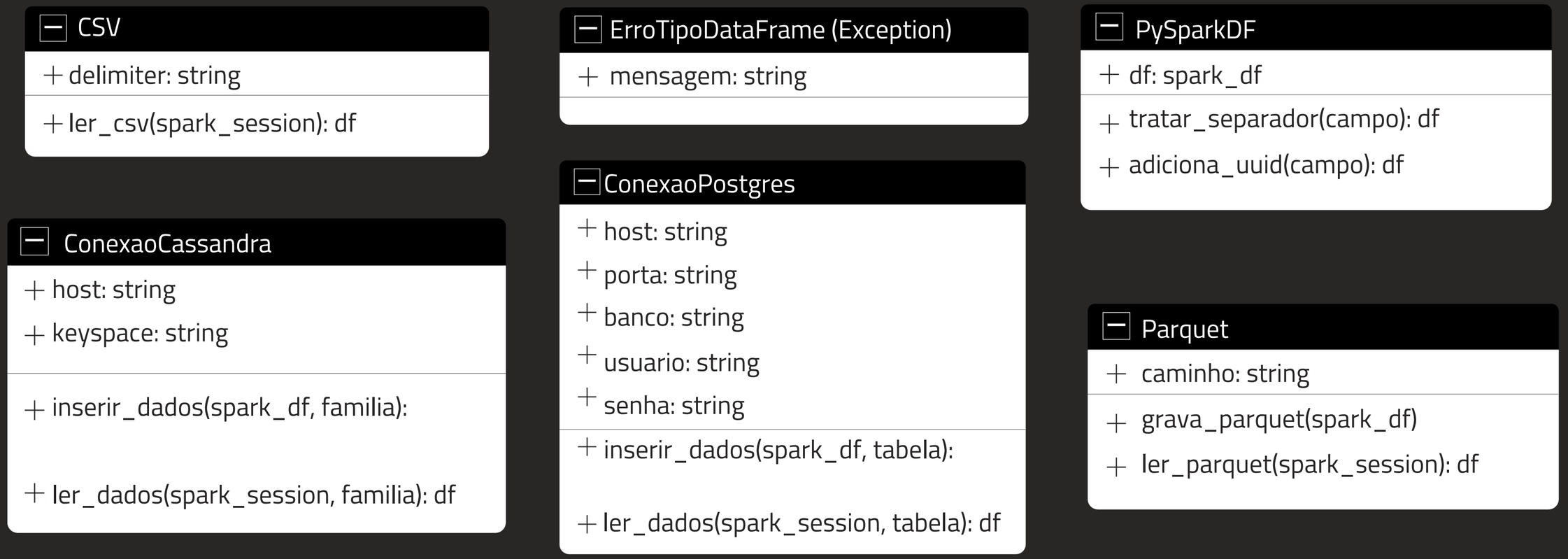
Ler dados Postgres

Inserir dados Cassandra

Ler dados Cassandra

Diagrama

Classes



Estrutura de Diretórios

Extração Datalake

modules/CSV.py
mineracao/job_1.py

```python
class CSV:
    def __init__(self, delimiter, encoding, caminho, schema):
        self.delimiter, self.encoding, self.caminho, self.schema = delimiter, encoding, caminho, schema

    def ler_csv(self, spark):
        try:
            df = spark.read.format("csv").option("header", "true")\
                .option("interSchema", "false")\
                .option("delimiter", self.delimiter)\
                ...
```

```python
try:
    df_arrecadacao = CSV(";", "ISO-8859-1", "gs://csv/arrecadacao.csv", schema_arrecadacao).ler_csv(spark)
    df_autuacao = CSV(";", "ISO-8859-1", "gs://csv/autuacao.csv", schema_autuacao).ler_csv(spark)
    df_barragens = CSV(";", "ISO-8859-1", "gs://csv/barragens.csv", schema_barragens).ler_csv(spark)
    df_beneficiada = CSV(",", "ISO-8859-1", "gs://csv/beneficiada.csv", schema_beneficiada).ler_csv(spark)
    ...
```

Tratamento

modules/PySparkDF.py

```
class PySparkDF:
    def __init__(self, df):
        if str(type(df)) == "<class 'pyspark.sql.dataframe.DataFrame'>":
            self.df = df
        else:
            raise ErroTypeDataframe()

    def trata_separador(self, campo):
        try:
            self.df = self.df.withColumn(campo, regexp_replace(campo, "," , "."))
        except Exception as e:
            print(str(e))
        else:
            return self.df

    def adiciona_uuid(self, campo):
        try:
            self.df = self.df.withColumn(campo, expr("uuid()"))
        except Exception as e:
            print(str(e))
        else:
            return self.df
    ...
```

Carga Base Relacional

modules/Conexao.py

```
1  class ConexaoPostgres:
2
3      def __init__(self, host, banco, usuario, senha):
4          self.host, self.banco, self.usuario, self.senha = host, banco, usuario, senha
5
6      def inserir_dados(self,df,tabela):
7          try:
8              df.write \
9                  .format("jdbc") \
10                 .option("url", f"jdbc:postgresql://{self.host}:5432/{self.banco}") \
11                 .option("numPartitions", "4") \
12                 .option("fetchsize", "5000") \
13                 .option("dbtable", tabela) \
14                 .option("driver", "org.postgresql.Driver") \
15                 .option("user", self.usuario) \
16                 .option("password", self.senha) \
17                 .mode('append')\
18                 .save()
19         except Exception as e:
20             print(str(e))
       ...
```

Carga Base Relacional

modules/Conexao.py

```
1  class ConexaoPostgres:
2
3      def __init__(self, host, banco, usuario, senha):
4          self.host, self.banco, self.usuario, self.senha = host, banco, usuario, senha
5
6      def inserir_dados(self,df,tabela):
7          try:
8              df.write \
9                  .format("jdbc") \
10                 .option("url", f"jdbc:postgresql://{self.host}:5432/{self.banco}") \
11                 .option("numPartitions", "4") \
12                 .option("fetchsize", "5000") \
13                 .option("dbtable", tabela) \
14                 .option("driver", "org.postgresql.Driver") \
15                 .option("user", self.usuario) \
16                 .option("password", self.senha) \
17                 .mode('append')\
18                 .save()
19         except Exception as e:
20             print(str(e))
       ...
```

Temp tables

```
1 create trigger before_trg_log
2     before insert on registro_log
3         for each row
4             execute procedure fn_cria_temp_table();
```

```
5 create or replace function fn_cria_temp_table() returns trigger as $$
6     begin
7         if(new.descricao_log = 'Criação temp tables') then
8             call pd_autuacao_cria_temp_table();
9             drop table if exists dm_autuacao;
10            create table dm_autuacao as (select * from tmp_autuacao);
  ....
```

```
11 create procedure pd_autuacao_cria_temp_table() as $$
12     drop table if exists tmp_autuacao;
13       CREATE TEMP TABLE IF NOT EXISTS tmp_autuacao AS select processo_cobranca,
14 tipo_pf_pj, nome_titular, substancia, municipio, uf, valor from autuacao;
15 $$ language sql;
```

Carga noSQL

mineracao/job_4.py

```
spark = SparkSession\
    .builder\
    .appName("job_4")\
    .config("spark.cassandra.connection.host","10.140.0.2") \
    .config("spark.cassandra.connection.port","9042") \
    .config("spark.cassandra.output.concurrent.writes", 500) \
    .config("spark.cassandra.output.throughputMBPerSec", "128") \
    .config("spark.cassandra.connection.keepAliveMS", "30000") \
    ...
```

```
try:
    conexao = ConexaoCassandra("mineracao")
    conexao.inserir_dados(df_arrecadacao, "arrecadacao")
    conexao.inserir_dados(df_autuacao, "autuacao")
    conexao.inserir_dados(df_barragens, "barragens")
    conexao.inserir_dados(df_beneficiada, "beneficiada")
    conexao.inserir_dados(df_distribuicao, "distribuicao")
    conexao.inserir_dados(df_municipio, "municipio")
    conexao.inserir_dados(df_pib, "pib")
    ...
```

Análises


```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  df_estado = pd.read_csv("sandi_rayssa_estados_ferro.csv")
5  df_estado.columns = ['UF', 'Total']
6
7  uf = df_estado['UF']
8  total = df_estado['Total']
9  fig, ax = plt.subplots(figsize =(12, 8))
10 plt.bar(uf, total, color="blue")
   ax.set_title("Estados possuem Mineração de Ferro - Barragens")
   plt.show()
```

Testes unitários


```python
import unittest

class TesteFuncoes(unittest.TestCase):
    def teste_grava_log(self):
    self.assertEqual(gravar_log("Testando log"), True)

```

```
root@debian:~/soulcode/final/modules/jobs$ python3 -m unittest
funcoes.TesteFuncoes
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

Orquestração

Orquestração


```
#!/bin/bash
echo "Criando cluster..."
gcloud dataproc clusters create mineracao --region=us-central1 --single-node \ --project fluid-vector-335716

echo "Job 1..."
gcloud dataproc jobs submit pyspark --cluster mineracao --region=us-central1 \
 --py-files gs://scripts/modules.zip gs://scripts/job_1.py

echo "Job 2..."
gcloud dataproc jobs submit pyspark --cluster mineracao --region=us-central1 \
 --py-files gs://scripts/modules.zip gs://scripts/job_2.py --jars=gs://scripts/postgresql-42.3.1.jar

...

echo "Job 5..."
gcloud dataproc jobs submit pyspark --cluster mineracao --region=us-central1 \
 --py-files gs://scripts/modules.zip gs://scripts/job_5.py \
 --properties 'spark.jars.packages=com.datastax.spark:spark-cassandra-connector_2.12:3.1.0'

echo "Excluindo cluster..."
gcloud dataproc clusters delete mineracao --region=us-central1 --quiet
```

Orquestração

Crontab

```
1   10 0 * * * /bin/bash pipeline.sh
    ...
```

```
2   Sun 23 Jan 022 00:10:01 AM UTC
3
4   Criando cluster...
5   Waiting on operation [projects/igneous-spanner-338620/regions/us-central]...
6   Waiting for cluster creation operation...
7   Waiting for cluster creation operation...done.
8   Created [https://dataproc.googleapis.com/V1/projetcts/igneous-spanner-338620]
9
10  Job 1...
11  Job [321312dasjk289321kjd9092902121d] submitted.
12  Waiting for job output...


    ...
```

Avaliação

Cloud Computing

Avaliação

Cloud Computing

Avaliação

Cloud Computing

=

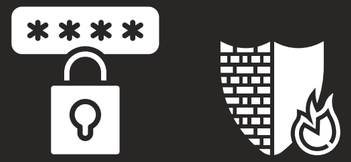
Avaliação

Cloud Computing

Economia de 92%

Segurança

Criptografia



Escalabilidade

Praticidade

Economia

Cluster

R$200 / dia

Pipeline `<linha de comando>`

`jobs`

24 h → 30 min

R$200 → R$15

Avaliação

ETL

Extrair | Tranformar | Carregar | Data Mart

ELT

Extrair | Carregar | Tranformar

Avaliação

Algorítmos

Análise

Processamento
e escalabilidade

Avaliação

Agradecimentos