

# Big Data – Data Engineering

**Tema da aula:**  
Spark



## BUSINESS SCHOOL

Graduação, pós-graduação, MBA, Pós-MBA, Mestrado Profissional, Curso In Company e EAD



## CONSULTING

Consultoria personalizada que oferece soluções baseadas em seu problema de negócio



## RESEARCH

Atualização dos conhecimentos e do material didático oferecidos nas atividades de ensino



Líder em Educação Executiva, referência de ensino nos cursos de graduação, pós-graduação e MBA, tendo excelência nos programas de educação. Uma das principais **escolas de negócio do mundo**, possuindo convênios internacionais com Universidades nos EUA, Europa e Ásia. +8.000 **projetos de consultorias** em organizações públicas e privadas.



Único curso de graduação em administração a receber as notas máximas



A primeira escola brasileira a ser finalista da maior competição de MBA do mundo



Única *Business School* brasileira a figurar no *ranking* LATAM



Signatária do Pacto Global da ONU



Membro fundador da ANAMBA - Associação Nacional MBAs



Credenciada pela AMBA - Association of MBAs



Credenciada ao Executive MBA Council



Filiada a AACSB - Association to Advance Collegiate Schools of Business



Filiada a EFMD - European Foundation for Management Development



Referência em cursos de MBA nas principais mídias de circulação



O **Laboratório de Análise de Dados** – LABDATA é um Centro de Excelência que atua nas áreas de ensino, pesquisa e consultoria em análise de informação utilizando técnicas de **Big Data, Analytics** e **Inteligência Artificial**.



Profª Drª Alessandra Montini

O LABDATA é um dos pioneiros no lançamento dos cursos de *Big Data* e *Analytics* no Brasil. Os diretores foram professores de grandes especialistas do mercado.

- +10 anos de atuação.
- +9000 alunos formados.

## Docentes

- Sólida formação acadêmica: doutores e mestres em sua maioria;
- Larga experiência de mercado na resolução de *cases*;
- Participação em congressos nacionais e internacionais;
- Professor assistente que acompanha o aluno durante todo o curso.

## Estrutura

- 100% das aulas realizadas em laboratórios;
- Computadores para uso individual durante as aulas;
- 5 laboratórios de alta qualidade (investimento +R\$2MM);
- 2 unidades próximas à estação de metrô (com estacionamento).



## PROF.ª DR.ª ALESSANDRA DE ÁVILA MONTINI

Diretora do LABDATA-FIA, apaixonada por dados e pela arte de lecionar. Tem muito orgulho de ter criado na FIA cinco laboratórios para as aulas de Big Data e Inteligência Artificial. Possui mais de 20 anos de trajetória nas áreas de Data Mining, Big Data, Inteligência Artificial e Analytics. Cientista de dados com carreira realizada na Universidade de São Paulo. Graduada e mestra em Estatística Aplicada pelo IME-USP e doutora pela FEA-USP. Com muita dedicação chegou ao cargo de professora e pesquisadora na FEA-USP, ganhou mais de 30 prêmios de excelência acadêmica pela FEA-USP e mais de 30 prêmios de excelência acadêmica como professora dos cursos de MBA da FIA. Orienta alunos de mestrado e de doutorado na FEA-USP. Parecerista da FAPESP e colunista de grandes portais de tecnologia.





# Conteúdo Programático

5



## ANALYTICS



ANALYTICS



ESTATÍSTICA  
APLICADA



MACHINE  
LEARNING



DEEP LEARNING

## TECNOLOGIA



CLOUD



HADOOP



SPARK



INGESTÃO DE  
DADOS



NoSQL

## LINGUAGEM



PYTHON

## PROJETO



PROJETO ANALYTICS



PROJETO FINAL



## Amauri Santos

Head de Dados na Genial Investimentos, Especialista em Engenharia de Dados e Software, 18 anos no mercado de TI



[amauri.tisoft@gmail.com](mailto:amauri.tisoft@gmail.com)



[+55 \(11\) 9 8467-4714](tel:+5511984674714)



<https://www.linkedin.com/in/amauri-santos-b565bb63>



# Conteúdo da Aula

- 1. Introdução
- 2. Conceito do Spark
  - i. Definição do Spark
  - ii. Fundamentos de Big data para o Spark
  - iii. Mercado
  - iv. Vantagens e Utilização
  - v. Api Core
  - vi. RDD
  - vii. Particionamento
  - viii. Parquet
  - ix. Delta Lake
  - x. Aplicações
  - xi. Ambientes e concorrentes
  - xii. DataBricks
  - xiii. Análise de Dados
  - xiv. Código e boas praticas
  - xv. Referências Bibliográficas



## SPARK

**Apache Spark** é uma plataforma de computação e análise de dados em grande escala que foi projetada para ser **rápida**, de **propósito geral** e relativamente simples de ser utilizada.







## Spark

O Spark ficou bastante famoso por ser uma forma de processamento de Big Data superior ao Hadoop Map Reduce. Map Reduce é um modelo de framework para processar grandes volumes de dados em paralelo, dividindo o trabalho em um conjunto de tarefas independentes. O diagrama abaixo ilustra de forma simples e intuitiva o processamento de big data com Map Reduce ao contar a quantidade de palavras em um livro:



## Definição do Spark - Fluxo



Problema de Negócio

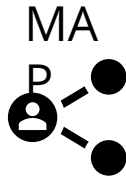
Origem dos Dados













Pipeline de Dados

Processamento e Resultado

Tomada de Decisão





			1652 palavras
			1968 palavras
			1847 palavras
			1509 palavras



**6976 palavras no total**





## SPARK

- Apesar do Map Reduce ser bem intuitivo, ele é bastante lento pois exige bastante leitura e escrita **dos dados em disco**. Essa característica é onde o Spark é melhor, pois ele faz um intercâmbio de dados entre o **disco e a memória ram**, permitindo assim que as operações aconteçam de forma mais rápida pois boa parte do processamento acontece na memória ao invés de acontecer no disco, como é o caso para o Map Reduce. Além dessas características, Spark também possui um otimizador de queries, um DAG scheduler e uma physical execution engine. Não entraremos em detalhes a respeito dessas outras features aqui. Para nosso propósito, saber que o Spark realiza processamento na memória ram é o mais importante.





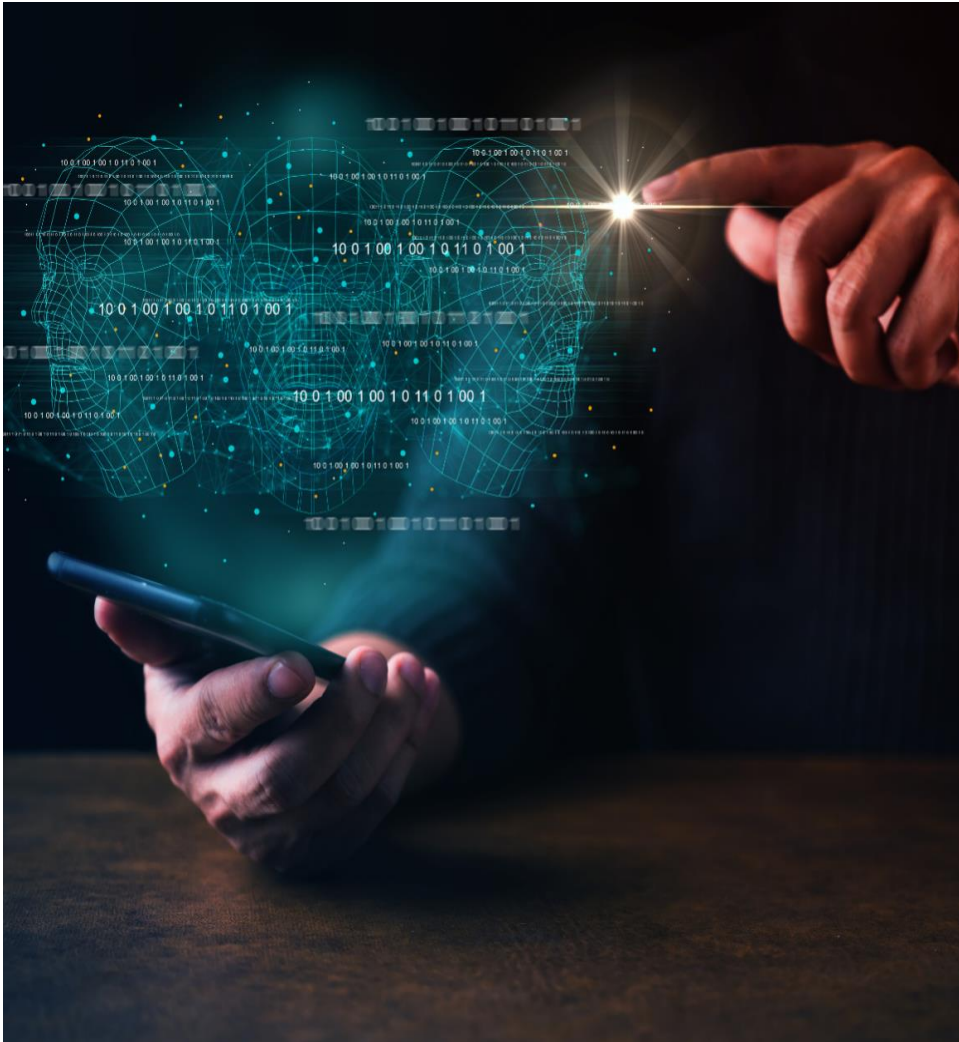


## Mercado

- Em empresas que utilizam o Databricks, o PySpark é bastante utilizado para criação de ETLs e pipelines de dados. Dessa forma, podemos pensar no PySpark como a ferramenta para criar as Analytical Base Tables e as também famosas OBT (One Big Table) tabelas que são utilizadas para treinar modelos de machine learning.
- Uma grande vantagem de se criar um ETL com PySpark é que a linguagem é totalmente escalável e não precisaremos mudar o nosso código caso o volume de dados cresça no futuro.
- 99,5% do Mercado usa o Spark para alguma solução.







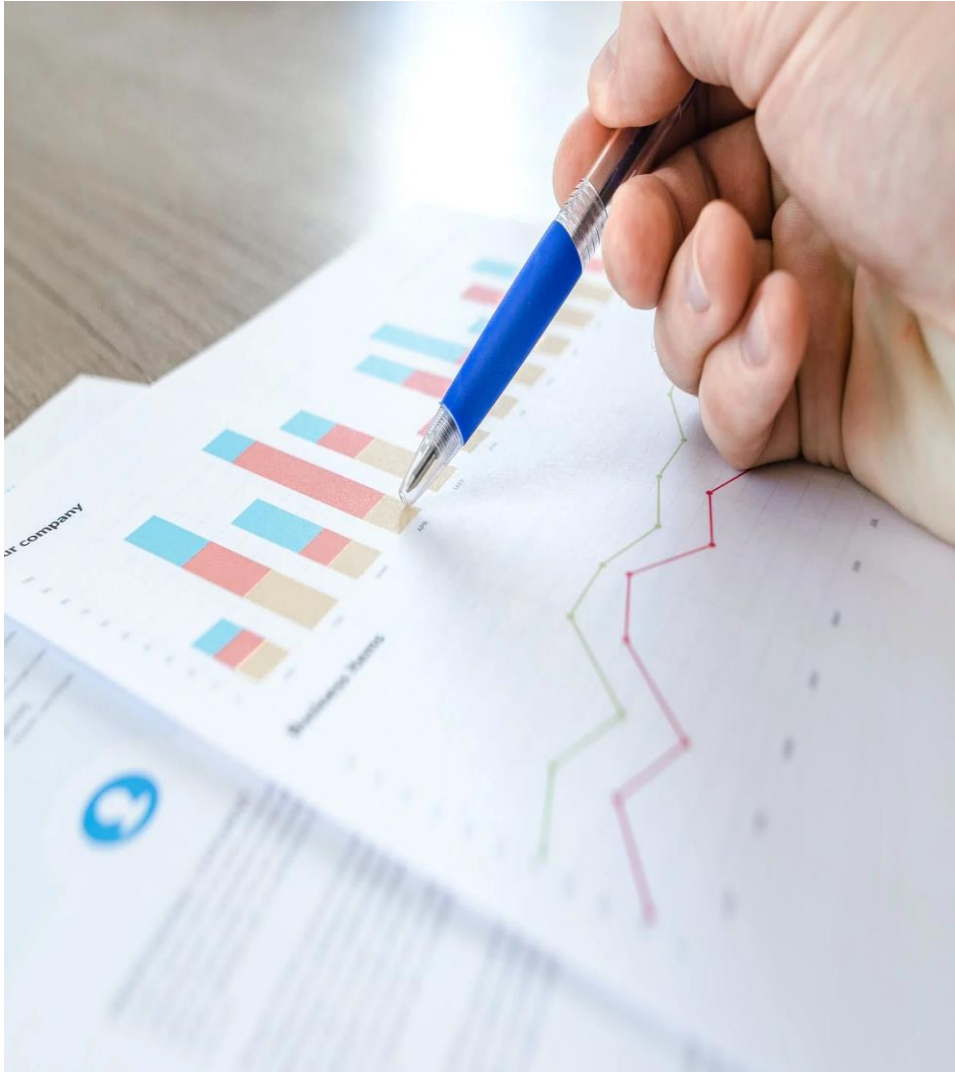
## Vantagens e Utilização

As vantagens para os negócios, lidar com o Spark requer uma combinação precisa de pessoas, processos e ferramentas. Alguns dos potenciais benefícios de negócios a partir da implementação do framework do Spark.

principais vantagens e utilizações:

- **Velocidade em processamento em Minutos**
- **Compatibilidade em qualquer ambiente**
- **Monitoramento em tempo real dos dados**
- **Capacidade de encontrar, adquirir, extrair, manipular, analisar, conectar, visualizar e disponibilizar**
- **Mitigação dos riscos na operação, sendo Negocio ou/e Infraestrutura**
- **Análise e interpretação dos comportamentos padrões com bases históricas e extremamente volumosas**





## SPARK – API CORE

- O Spark possui APIs fáceis de usar para operar em grandes conjuntos de dados. Isso inclui uma coleção de mais de 100 operadores para transformar dados e APIs familiares de quadro de dados para manipular dados semiestruturados.
- O Spark vem com bibliotecas de alto nível, incluindo suporte para consultas SQL, streaming de dados, aprendizado de máquina e processamento de gráficos. Essas bibliotecas padrão aumentam a produtividade do desenvolvedor e podem ser perfeitamente combinadas para criar fluxos de trabalho complexos.
- O Spark também é rápido quando os dados são armazenados em disco e atualmente detém o recorde mundial de classificação em disco em grande escala.



# SPARK

## 1. Conceito

16

Spark SQL +  
DataFrames

Streaming

MLlib  
*Machine Learning*

GraphX  
*Graph  
Computation*

Spark Core API

R

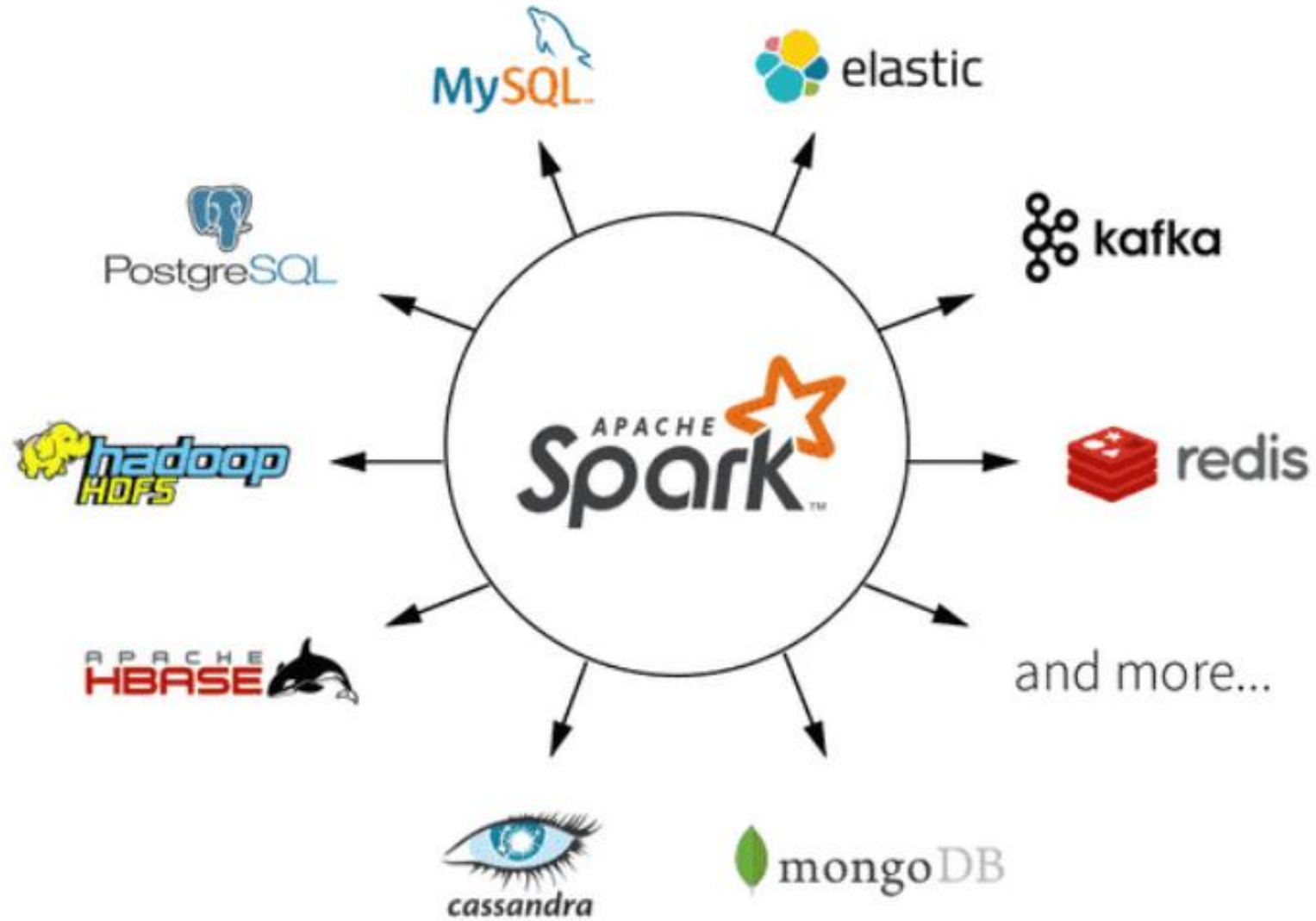
SQL

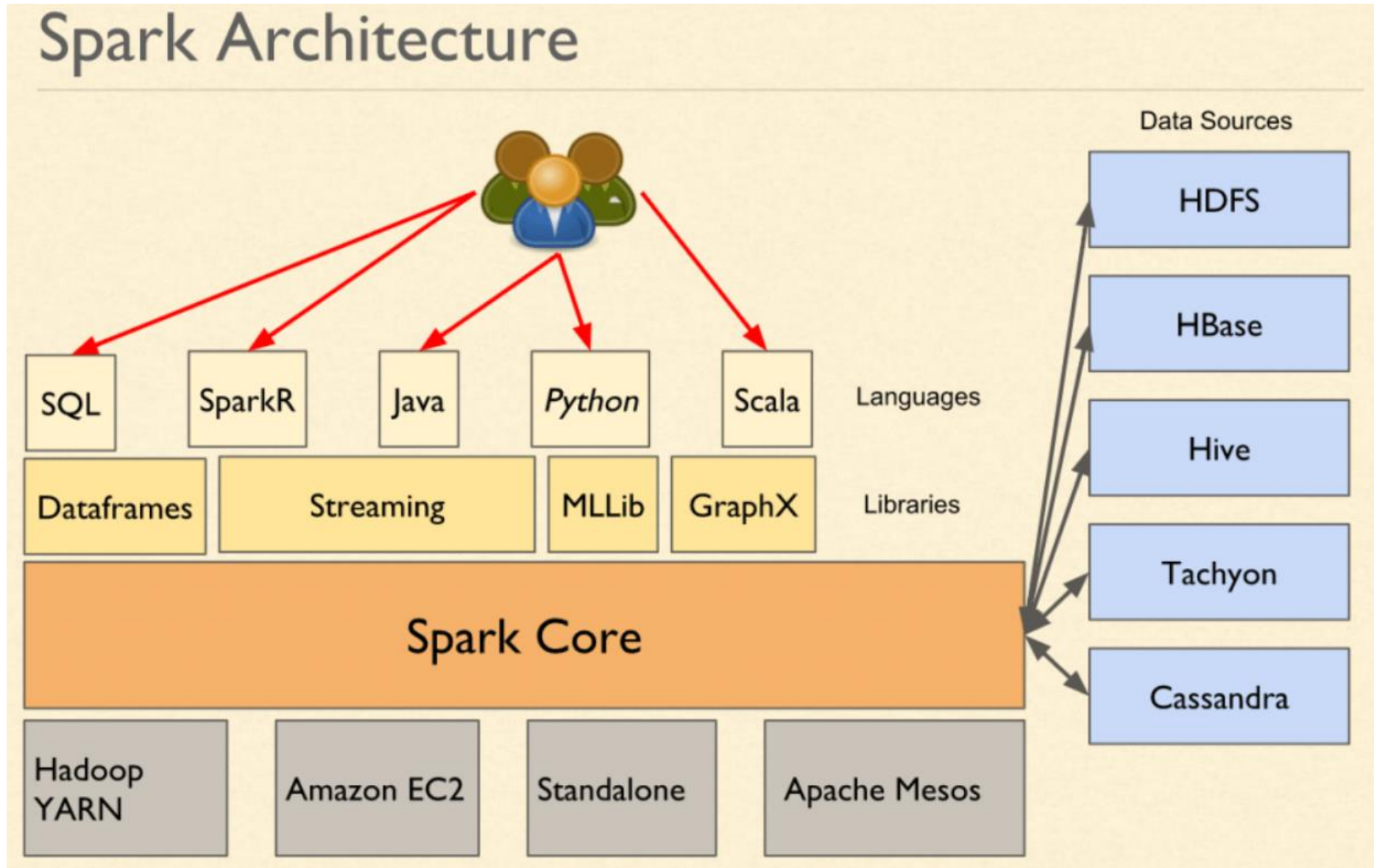
Python

Scala

Java











## Spark SQL

O Spark SQL é uma biblioteca poderosa que integrantes não técnicos de equipes, como por exemplo: Analistas de Negócio e Analistas de Dados, podem utilizar para realizar análise de dados para suas empresas, ou seja, utilizando o SQL tradicional, dando mais lateralidade e profundidade em todas as áreas e pessoas da empresa, sendo mais fácil democratizar o acesso aos dados.





## Spark Streaming

O Spark Streaming é um fluxo contínuo de entrada de dados que usa um fluxo discretizado chamado DStream. É possível criar um DStream a partir de fontes de entrada como os Hubs de Eventos ou o Kafka, ou seja, ter dados em tempo real o NRT (new real time), podemos ser utilizados em arquiteturas de evento (kappa) ou lambda (stream + batch), até mesmo servido dados em apis.





## Spark MLlib

MLlib é uma biblioteca Spark principal que fornece vários utilitários úteis para tarefas de aprendizado de máquina:

Estatística tradicional com Spark:

- a) Regressão Linear
- b) Análise de Cluster
- c) Regressão Logística
- d) Árvore de decisão



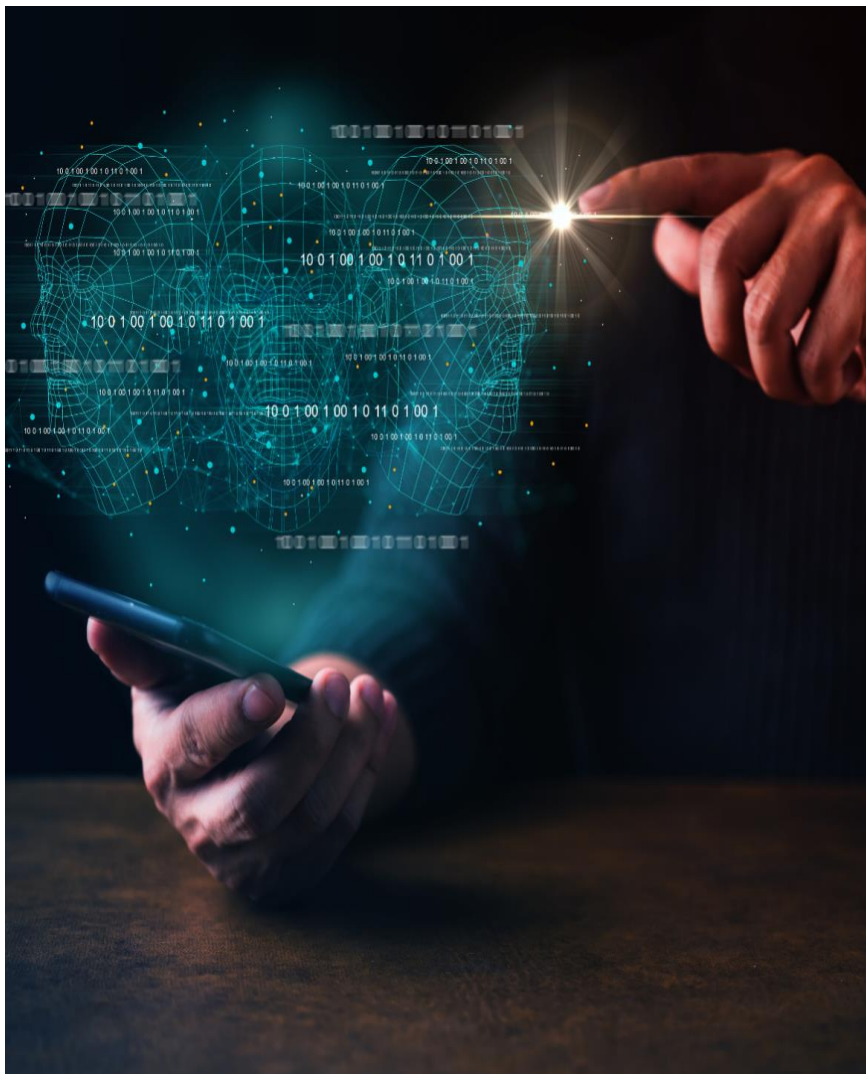


## Spark GraphX

O GraphX unifica ETL, análise exploratória e computação gráfica iterativa em um único sistema. Você pode visualizar os mesmos dados como gráficos e coleções, transformar e unificar gráficos com RDDs de forma eficiente e escrever algoritmos de gráficos iterativos personalizado, ou seja, trabalhamos com o sistema e algoritmos de grafos, que é muito usado em gps, sistemas de localização, logística e tráfego.





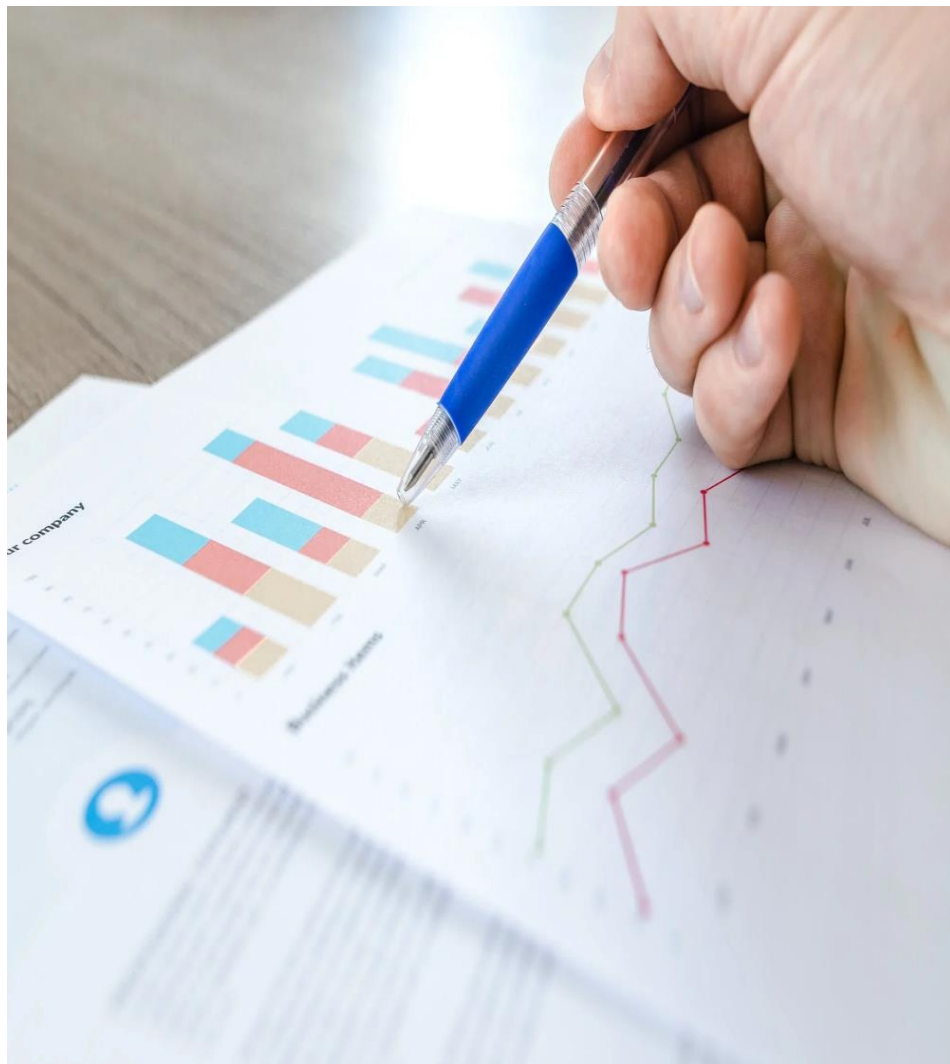


## Arquitetura do Spark

- O apache Spark é um mecanismo de computação unificado e desenvolvido ativamente é um conjunto de bibliotecas. Ele é usado para processamento paralelo de dados em clusters de computadores e se tornou uma framework padrão para qualquer desenvolvedor ou cientista de dados interessado em Big Data.
- O Spark oferece suporte a várias linguagens de programação amplamente usadas, como Java, Python , R e Scala. Ele inclui bibliotecas para uma ampla variedade de tarefas, como SQL, Streaming, ML, GX.
- Processamento de big data ou/e a uma escala incrivelmente grande.



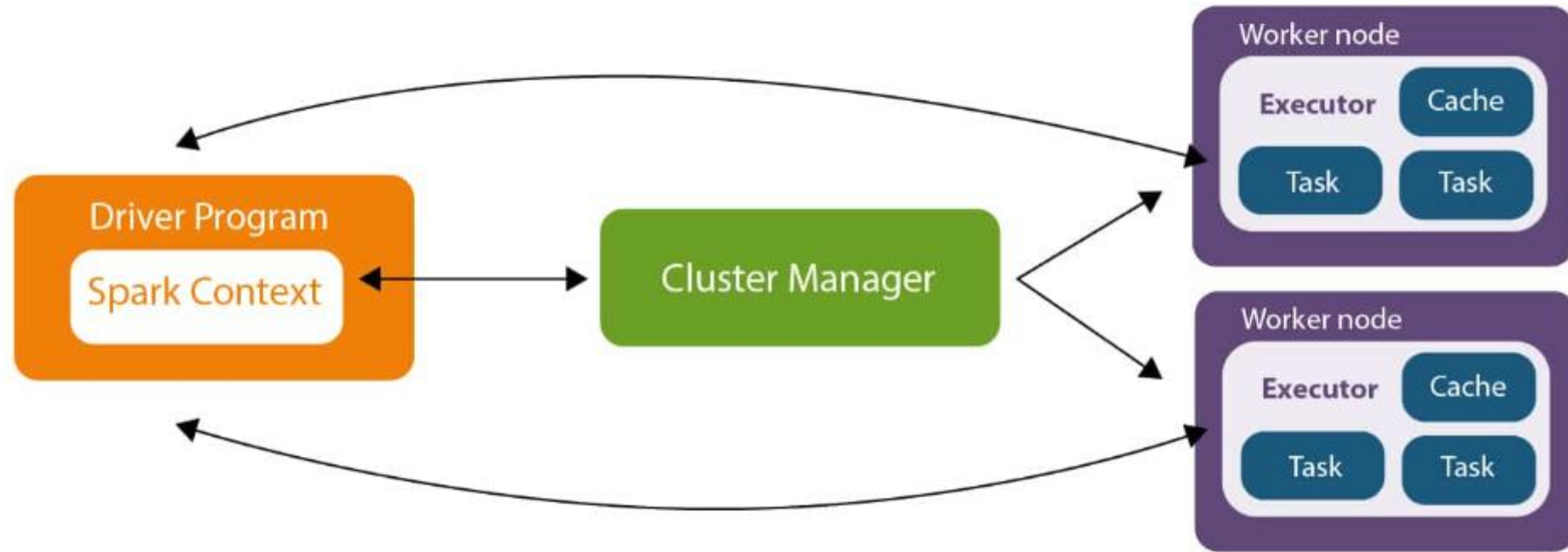




## Spark

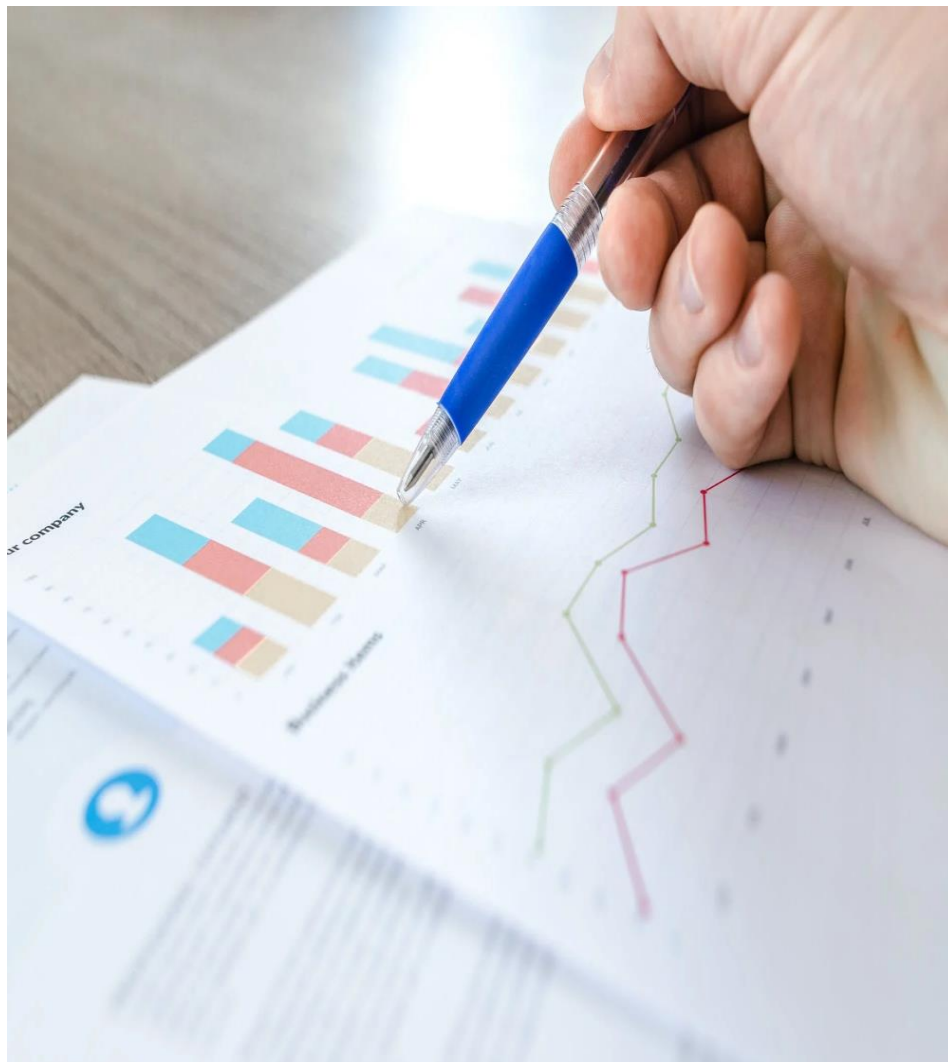
A estrutura do Apache Spark usa uma arquitetura **Principal – Secundário** (mestre-escravo) que consiste em um driver, que é executado como um nó Principal (mestre), e muitos executores que são executados como nós de trabalho no cluster. O Apache Spark também pode ser usado para processamento em lote e processamento em tempo real.





- Driver Program na arquitetura Apache Spark chama o programa principal de um aplicativo e cria SparkContext. Um SparkContext consiste em todas as funcionalidades básicas. O Spark Driver contém vários outros componentes, como DAG Scheduler, Task Scheduler, Backend Scheduler e Block Manager, que são responsáveis por traduzir o código escrito pelo usuário em tarefas que são realmente executadas no cluster.
- O Spark Driver e o SparkContext supervisionam coletivamente a execução do trabalho dentro do cluster. O Spark Driver trabalha com o Cluster Manager para gerenciar vários outros trabalhos. O gerenciador do cluster faz o trabalho de alocação de recursos. E então, o trabalho é dividido em várias tarefas menores que são distribuídas para nós de trabalho.
- Sempre que um RDD é criado no SparkContext, ele pode ser distribuído em muitos nós de trabalho e também pode ser armazenado em cache lá.
- Os nós de trabalho executam as tarefas atribuídas pelo Cluster Manager e as retornam ao Spark Context.
- Um executor é responsável pela execução dessas tarefas. O tempo de vida dos executores é o mesmo do Aplicativo Spark. Se quisermos aumentar o desempenho do sistema, podemos aumentar o número de trabalhadores para que os trabalhos possam ser divididos em porções mais lógicas.

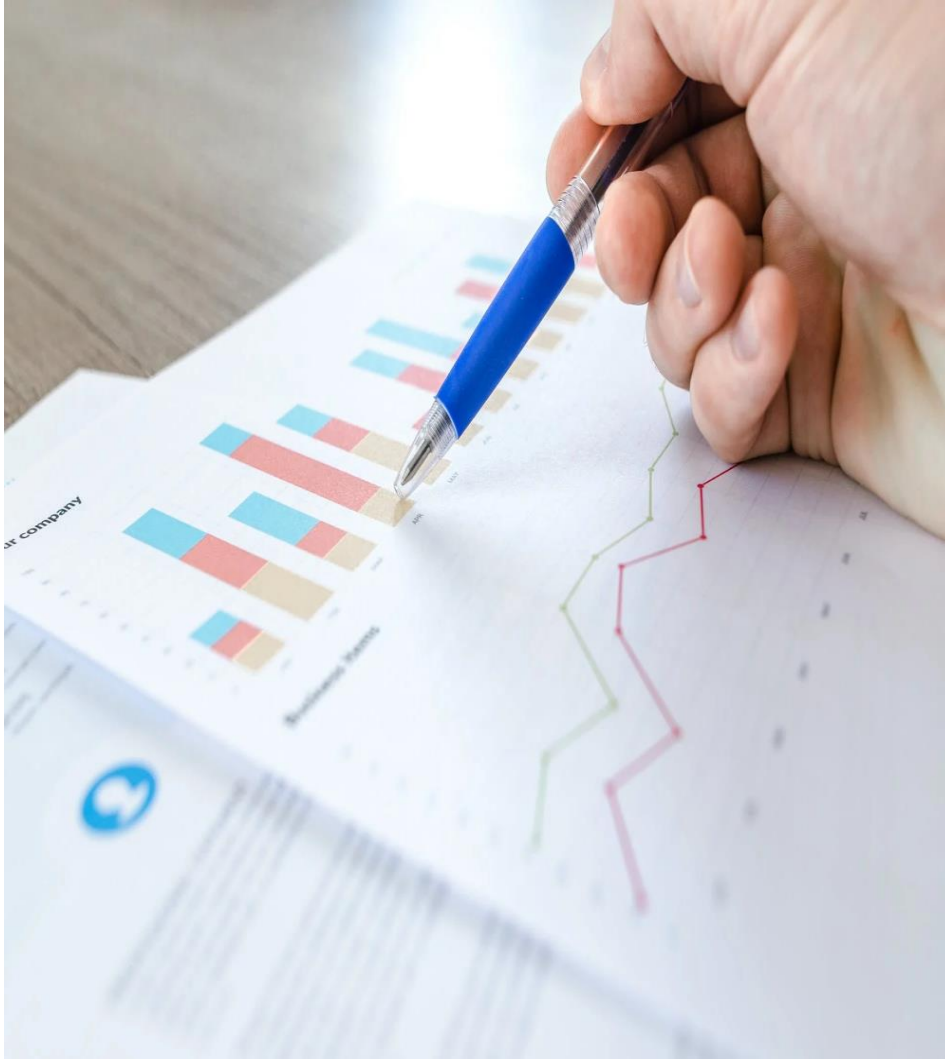




## Spark Driver

O Spark Driver é como o banco do motorista de um aplicativo Spark. Ele atua como o controlador da execução de um aplicativo Spark. O driver Spark mantém todos os estados do aplicativo em execução no cluster Spark. Para obter recursos físicos e executar executores, o driver Spark deve ter interface com o gerenciador de cluster.



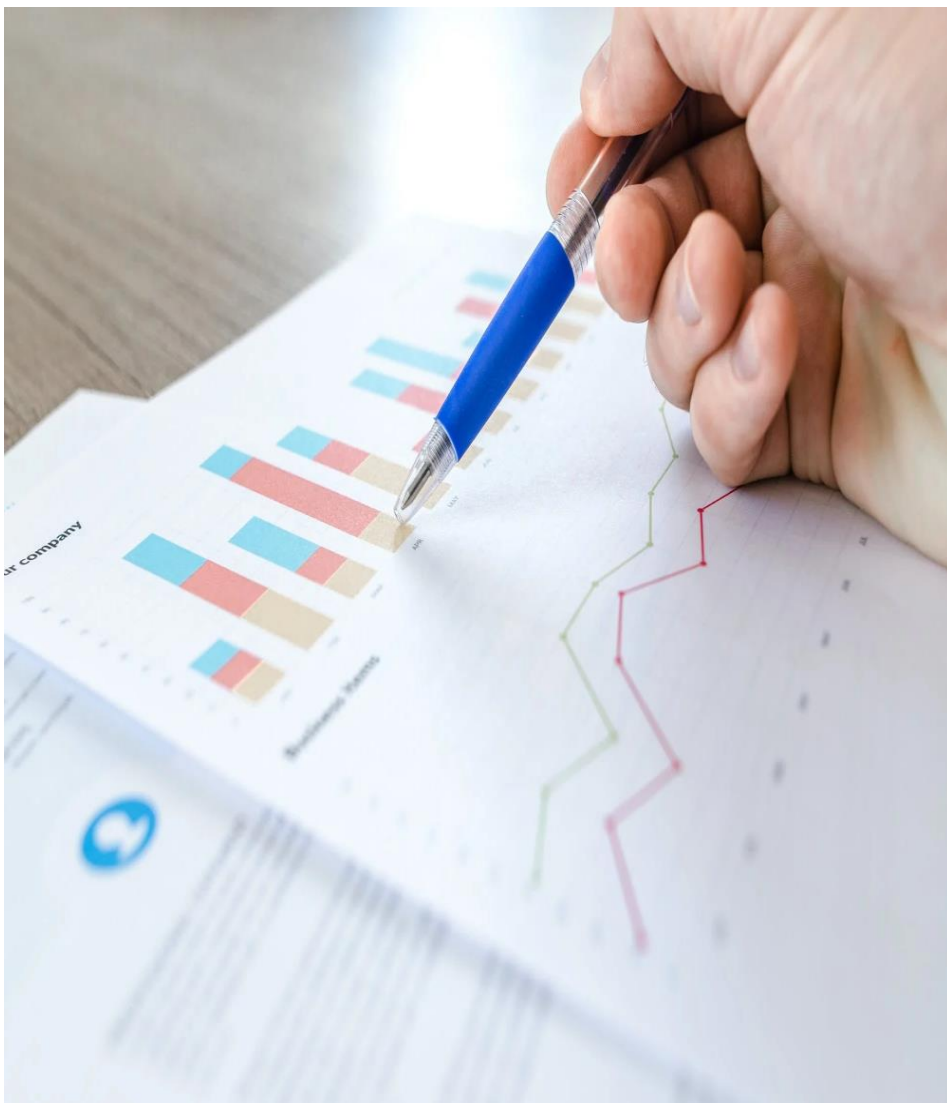


## Spark Executors

As tarefas atribuídas pelo driver Spark são executadas pelos executores Spark. A principal responsabilidade de um executor do Spark é pegar as tarefas atribuídas, executá-las e relatar seu estado de sucesso ou falha e resultados. Cada aplicativo Spark tem seus próprios processos executores separados.



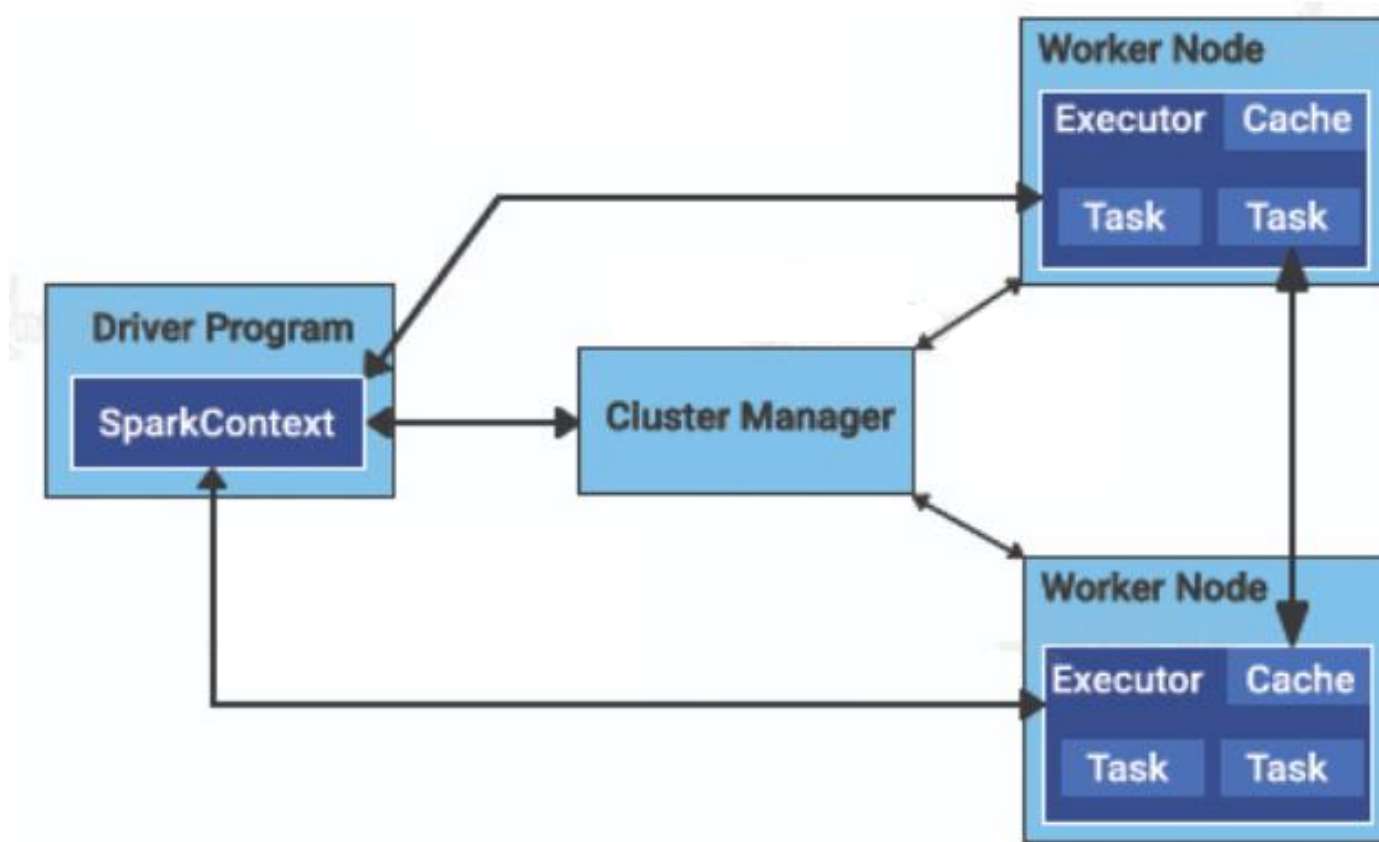




## Spark Cluster Manager

- O gerenciador de cluster mantém um cluster de máquinas que executarão aplicativos Spark. Ele tem seu próprio driver chamado abstrações “master” e “worker”. Eles estão vinculados a máquinas físicas em vez de processos como no Spark.
- A máquina, se você observar à esquerda da ilustração da arquitetura Spark, é o Cluster Manager Driver Node. Os círculos são os processos daemon que estão executando e gerenciando todos os nós de trabalho individuais. Estes são apenas os processos do Cluster Manager. Durante esse período, nenhum aplicativo Spark está em execução.
- Quando chega a hora de executar um aplicativo Spark, os recursos são solicitados ao gerenciador de cluster para executá-lo. Dependendo da configuração do aplicativo, pode ser um local para executar o driver Spark ou simplesmente recursos para os executores do aplicativo Spark.
- Ao longo da execução do aplicativo Spark, o Cluster Manager gerencia as máquinas subjacentes nas quais o aplicativo está sendo executado.







## Modes of Execution

- Um modelo de execução ajuda a determinar onde os recursos mencionados anteriormente estão localizados fisicamente quando o aplicativo é executado. Existem três modos de execução para escolher:
- Cluster Mode
- Client Mode
- Local Mode





## Cluster Mode

- O modo de cluster é a maneira mais comum de executar aplicativos Spark durante o qual um script Python, JAR ou R pré-compilado é enviado a um gerenciador de cluster por um usuário. O processo do driver é então iniciado em um nó do trabalhador dentro do cluster pelo gerenciador do cluster, além dos processos do executor. Isso implica que o gerenciador de cluster é responsável por manter todos os processos relacionados ao aplicativo Spark.







## Client Mode

- O modo cliente é quase o mesmo que o modo cluster, exceto que o driver Spark permanece na máquina cliente que enviou o aplicativo. Isso significa que a máquina cliente mantém o processo do driver Spark e o gerenciador do cluster mantém os executores. Essas máquinas são comumente conhecidas como máquinas de gateway ou nós de borda.







## Local Mode

- No modo local, todo o aplicativo Spark é executado em uma única máquina. Ele observa o paralelismo por meio de threads nessa única máquina. Essa é uma maneira comum de testar aplicativos ou experimentar o desenvolvimento local. No entanto, não é recomendado para executar aplicativos de produção.





## RDD

- O Apache Spark possui uma arquitetura de camadas bem definida, projetada em duas abstrações principais:
- Conjunto de dados distribuído resiliente (RDD): RDD é uma coleção fundamental imutável (somente leitura) de elementos ou itens que podem ser operados em vários dispositivos ao mesmo tempo (processamento paralelo massivo). Cada conjunto de dados em um RDD pode ser dividido em porções lógicas, que são executadas em diferentes nós de um cluster.
- Gráfico Acíclico Direcionado (DAG): DAG é a camada de agendamento da arquitetura Apache Spark que implementa agendamento orientado por estágio. Comparado ao MP, que cria um gráfico em dois estágios, Map e Reduce, o Apache Spark pode criar DAGs que contêm muitos estágios.





## RDD

- O Resilient Distributed Dataset (RDD) é o trunfo da tecnologia Spark, que é uma importante estrutura de dados distribuídos. Em várias máquinas, ele é particionado fisicamente dentro de um cluster e é uma entidade centralizada quando considerado logicamente. Dentro de um cluster, o embaralhamento de dados entre máquinas pode ser reduzido controlando como vários RDDs são coparticionados. Existe um operador 'partition-by' que, ao redistribuir os dados no RDD original, cria um novo RDD nas máquinas do cluster





## RDD

- O acesso rápido é o benefício óbvio quando o RDD é armazenado em cache de forma ideal na RAM. Atualmente, no mundo do Analytics, a granularidade do cache é feita no nível do RDD. É como tudo ou nada. Todo o RDD é armazenado em cache ou não é armazenado em cache. Se houver memória suficiente disponível no cluster, o Spark tentará armazenar em cache o RDD. Isso é feito com base no algoritmo de despejo menos usado recentemente (LRU). A expressão da lógica do aplicativo como uma sequência de transformações é possível por meio de uma estrutura de dados abstrata fornecida pelo RDD. Esse processo pode acontecer independentemente da natureza distribuída subjacente dos dados.
- As lógicas do aplicativo geralmente são expressas em transformação e ação. A dependência de processamento, DAG entre RDDs é o que 'transformação' especifica. O tipo de saída é especificado por 'ação'. Para descobrir a sequência de execução do DAG, o agendador executa uma classificação de topologia que remonta aos nós de origem. Este nó representa um RDD em cache.



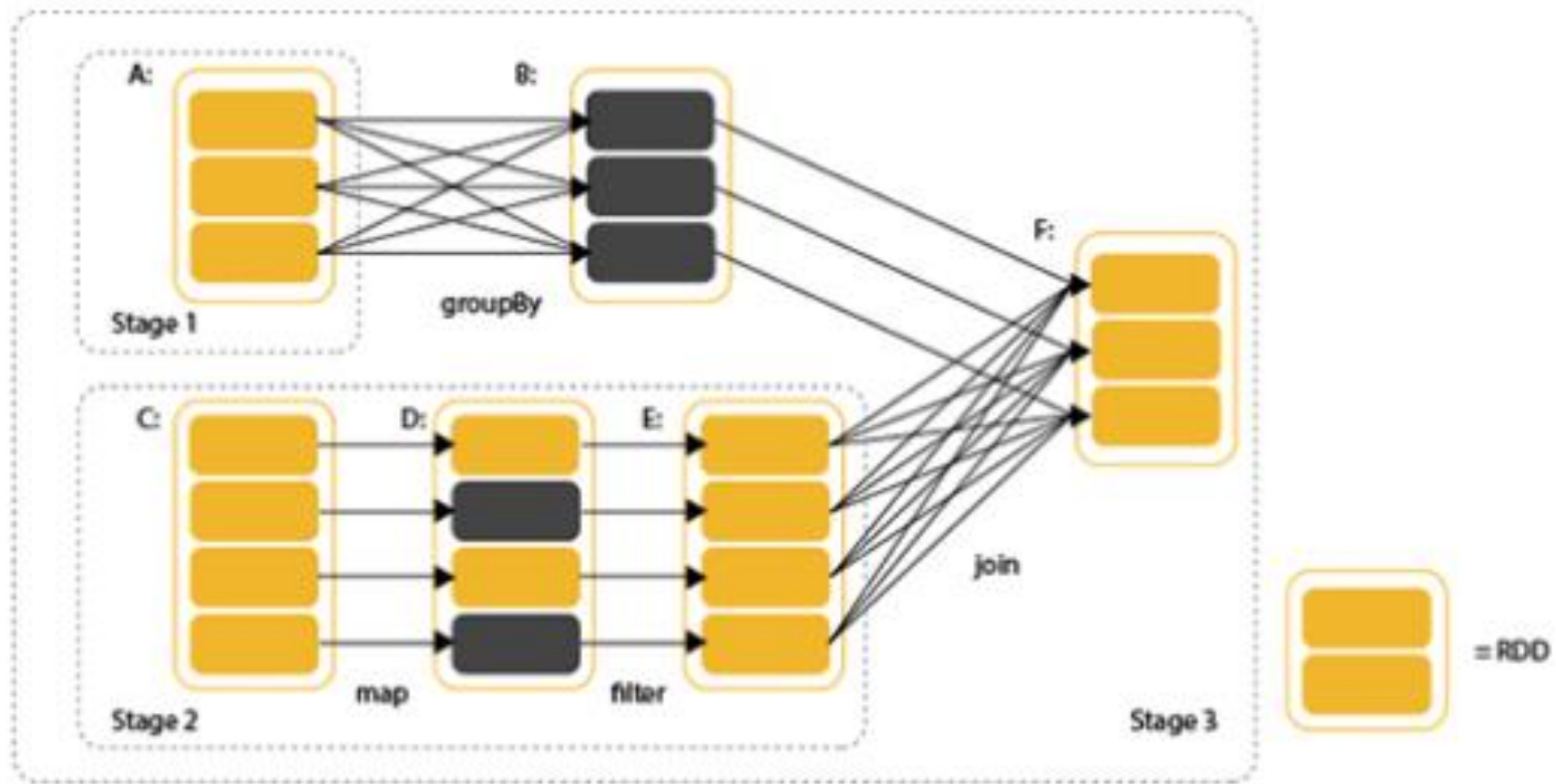
## RDD

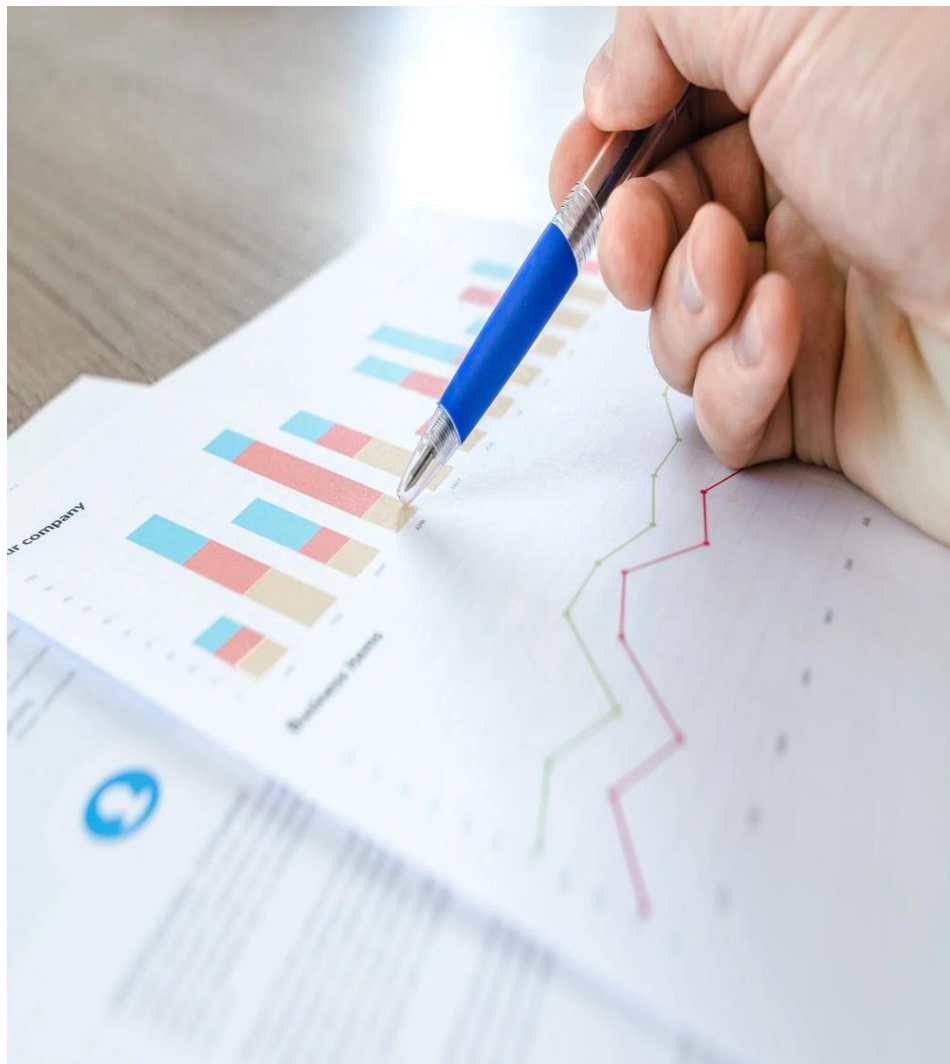
- As dependências do Spark serão de dois tipos, e são dependência estreita e dependência ampla.

Dependência Estreita	Ampla Dependência
A dependência estreita é onde um único RDD filho consome todas as partições do RDD pai.	Na dependência ampla, diferentes RDDs filhos obterão várias divisões do RDD pai com base nas chaves dos RDDs filhos.
Por exemplo, a dependência estreita pode ser comparada a uma família em que, para um dos pais (RDD), existe apenas um único filho (RDD). Neste caso, todos os bens e heranças (partições) irão para o filho único (RDD).	A ampla dependência pode ser comparada a outra família em que um único pai (RDD) tem vários filhos (RDD). Aqui, a riqueza dos pais (partição) seria dividida entre todos os filhos (RDD).









## Gerenciadores de Cluster

- Docker é um projeto opensource escrito em Go, que torna a o SparkContext pode trabalhar com vários gerenciadores de cluster, como Standalone Cluster Manager, YARN ou Spark UI, que alocam recursos para contêineres nos nós de trabalho. O trabalho é feito dentro desses contêineres.



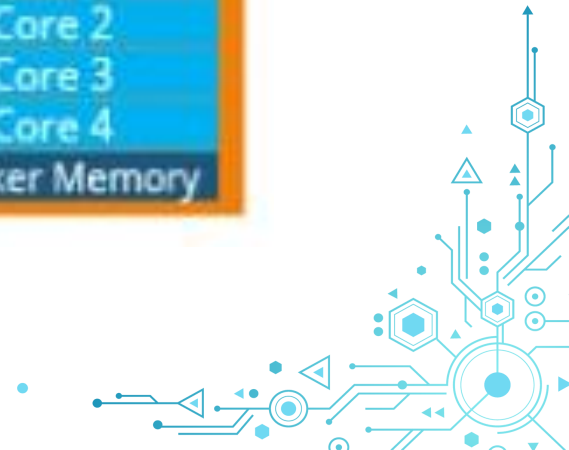
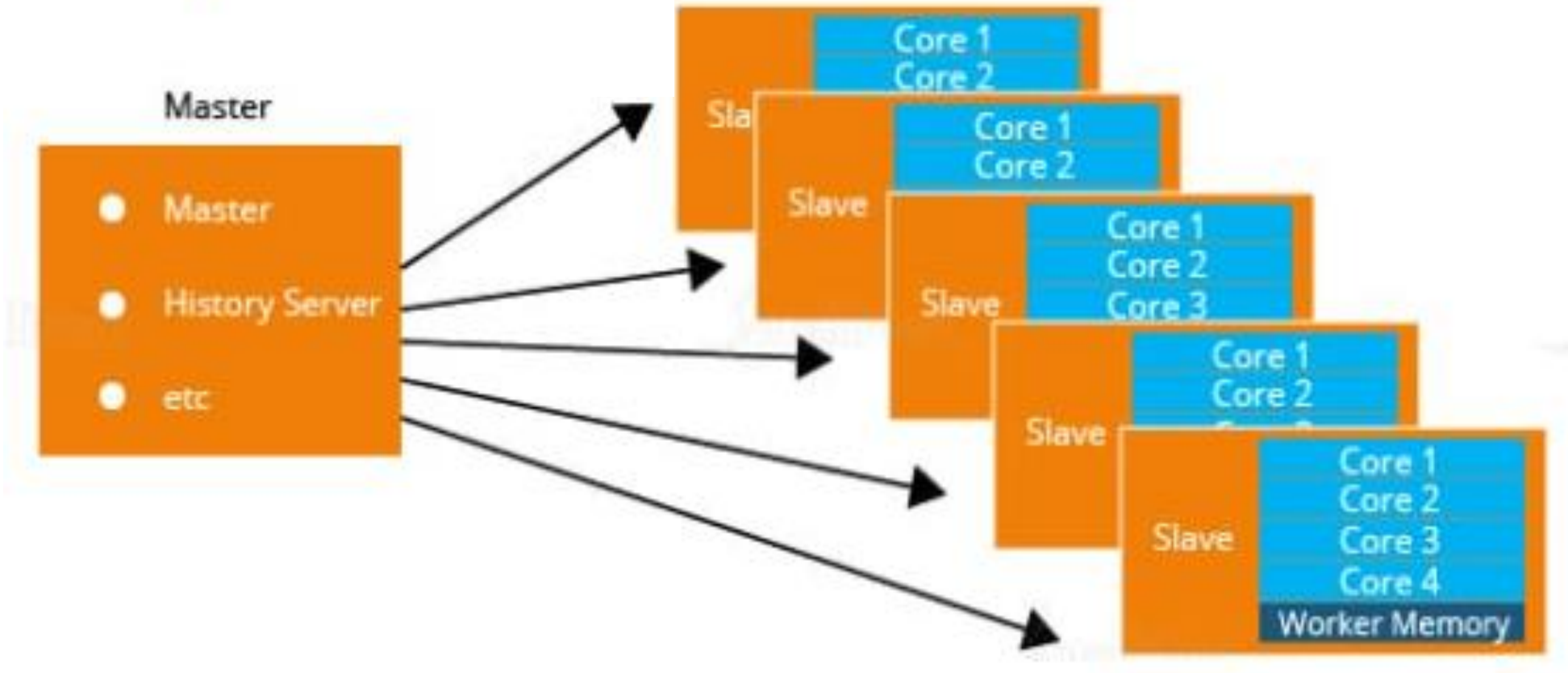


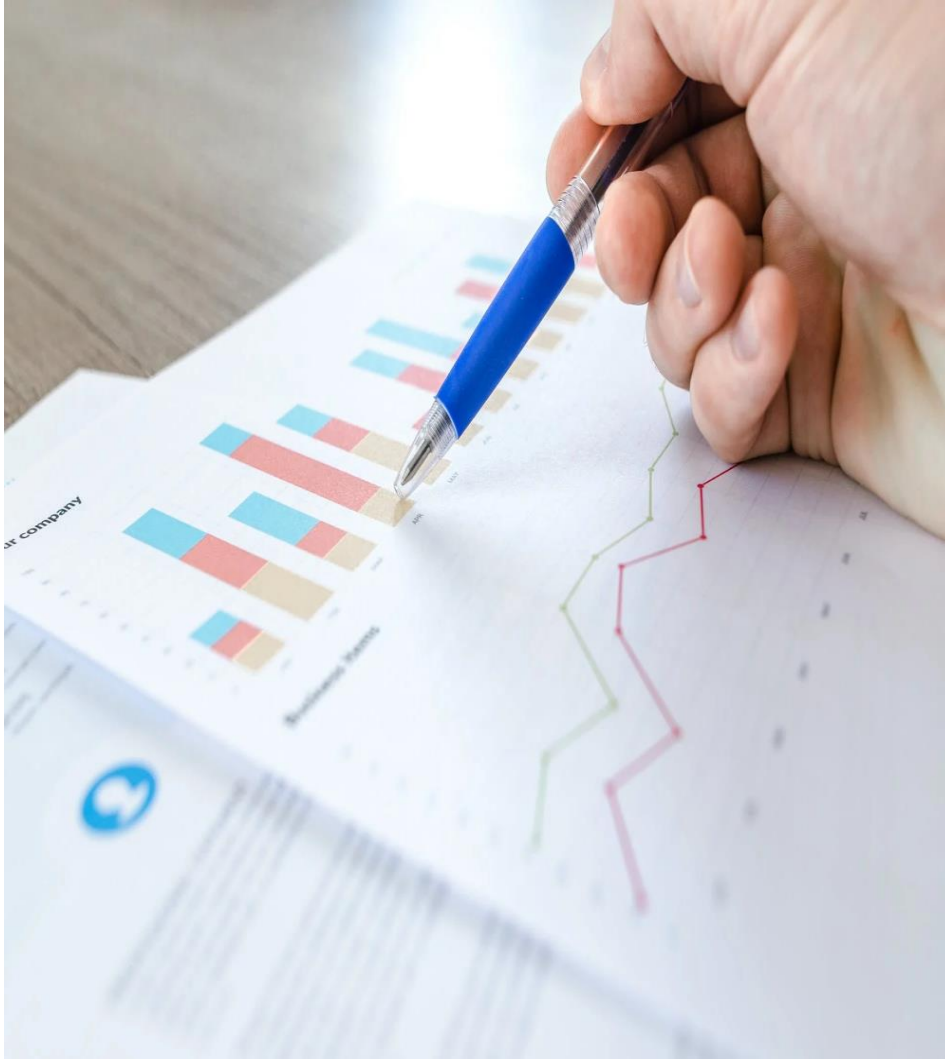
## Cluster autônomo

- O Standalone Principal é o Resource Manager e o Standalone Worker é o trabalhador no Spark Standalone Cluster.
- No modo Standalone Cluster, há apenas um executor para executar as tarefas em cada nó do trabalhador.
- Um cliente estabelece uma conexão com o Standalone Master, solicita recursos e inicia o processo de execução no nó do trabalhador.
- Aqui, o cliente é o Principal da aplicação e solicita os recursos do Resource Manager. Neste Cluster Manager, temos uma interface de usuário da Web para visualizar todos os clusters e estatísticas de trabalho.



# Spark Standalone Architecture





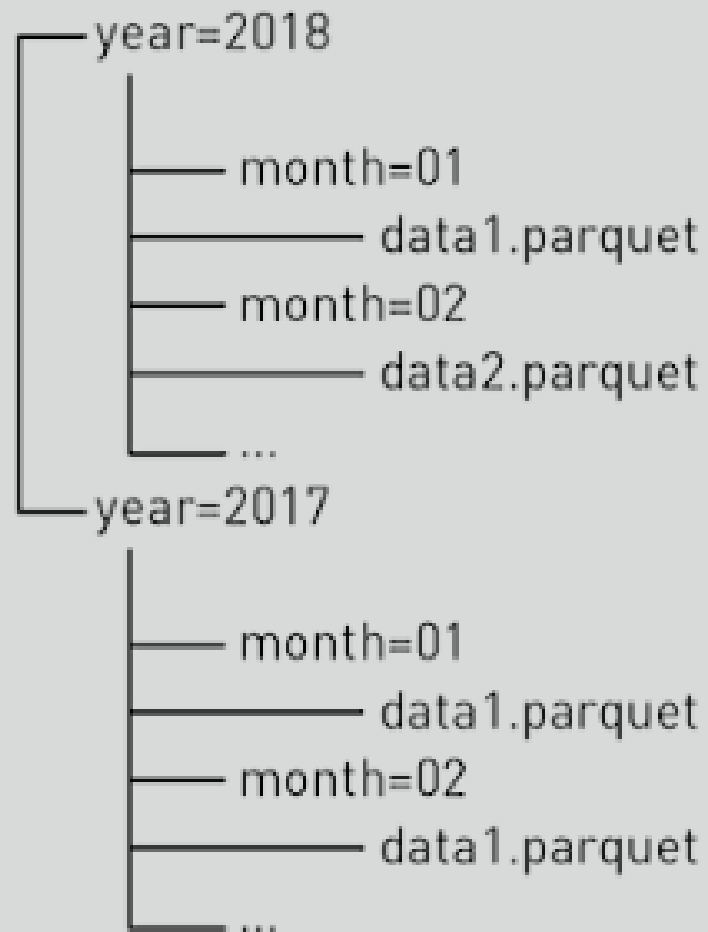
## Particionamento

- O conceito de adicionar informações(arquivos) em diferente diretórios, sendo que podemos ler todos ou apenas uma fatia do todo, ou seja, temos mais organização, velocidade na leitura e controle das informações novas e antigas, baseando-se no range o delta dos campos que representam a integridade dos dados e seu comportamento.

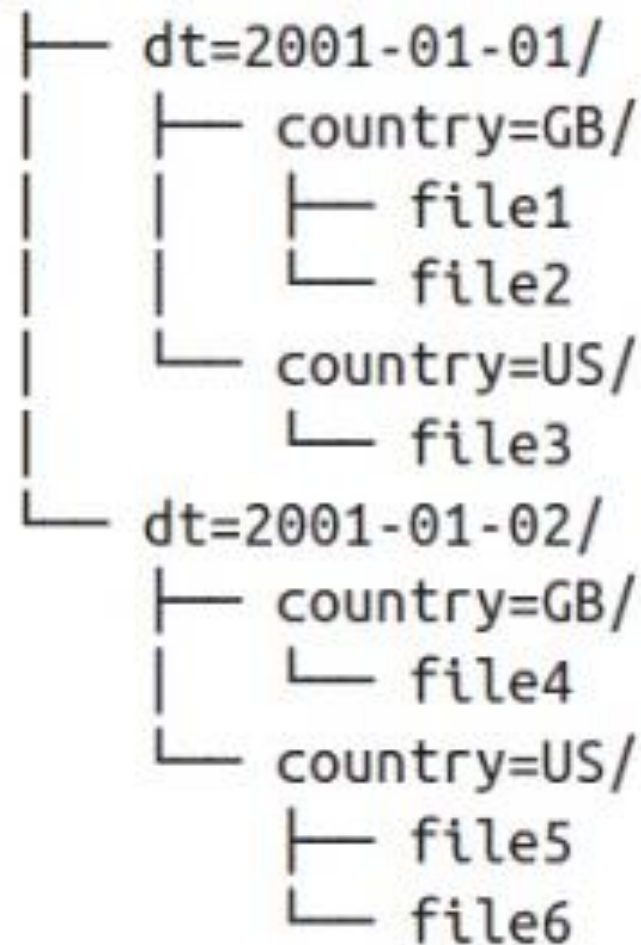


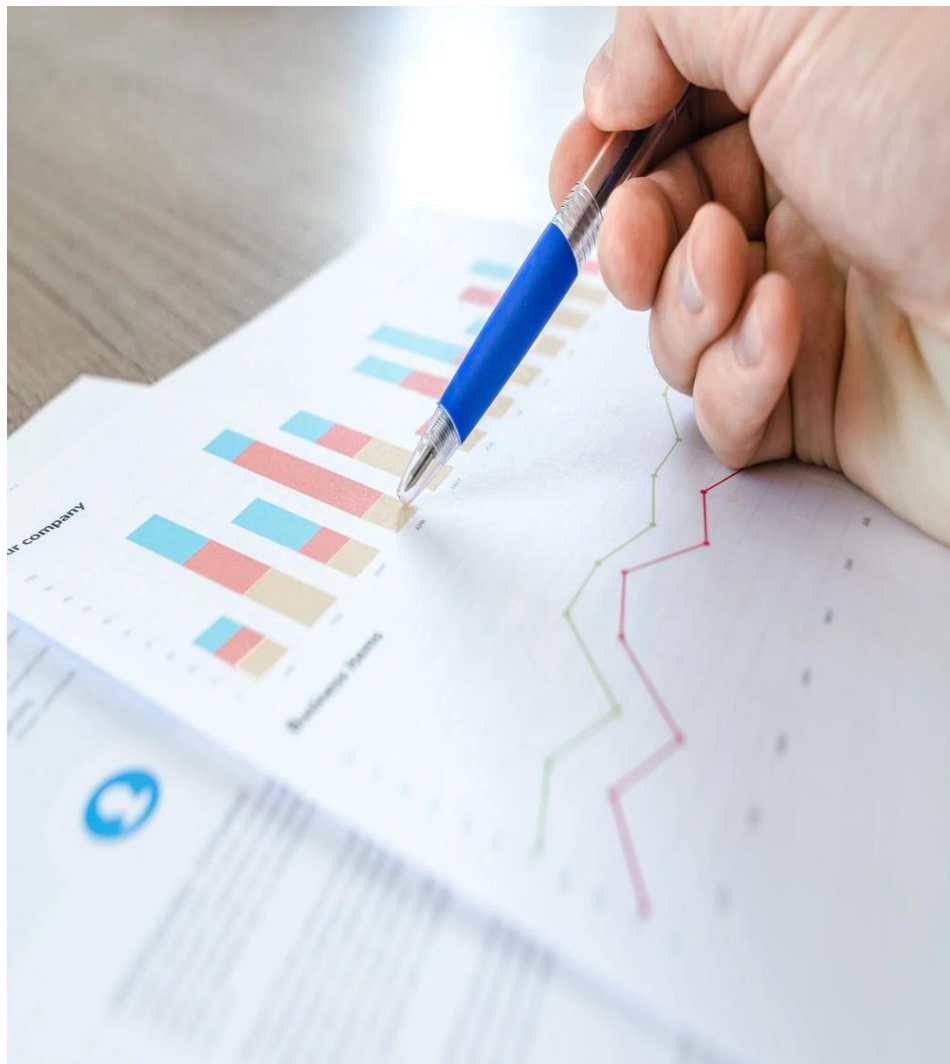


file directory



/user/hive/warehouse/logs





## Parquet

- Apache parquet é um formato de arquivo de código aberto que fornece armazenamento eficiente e velocidade de leitura rápida. Ele usa um formato de armazenamento híbrido que armazena sequencialmente pedaços de colunas, proporcionando alto desempenho ao selecionar e filtrar dados. Além do forte suporte ao algoritmo de compactação, ele também fornece alguns truques inteligentes para reduzir as varreduras de arquivo e codificar variáveis repetidas.



## Parquet

- Digamos que somos engenheiros de dados. Procuramos criar uma arquitetura de dados que facilite os Processos Analíticos Online ( OLAP ), que são apenas consultas seletivas voltadas para a análise de dados. Alguns exemplos de funções de dados otimizadas em um ambiente OLAP são análise exploratória de dados ou ciência da decisão.
- Velocidade de leitura : com que rapidez podemos acessar e decodificar informações relevantes de arquivos binários
- Tamanho no disco : quanto espaço nosso arquivo precisa quando em formato binário, compactação de 75% a 87% dependendo da versão que estiver usando



## Parquet

1. Baseada em linha (Row-based): armazena sequencialmente as linhas (CSV).
2. Com base em colunas (Column-based): armazena colunas sequencialmente (ORC).
3. Base híbrida (Hybrid-base): armazena sequencialmente pedaços de colunas (Parquet).

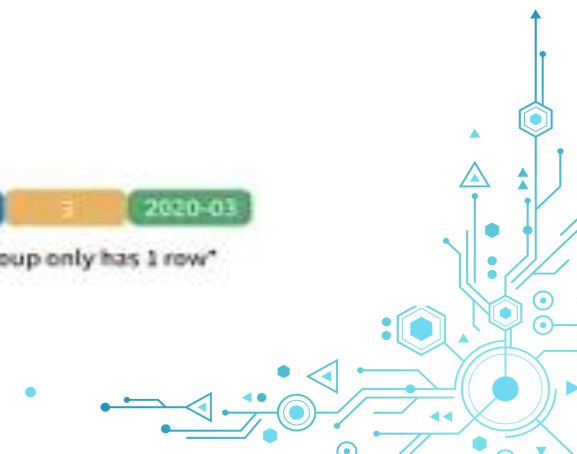
### Row-Based Storage Layout

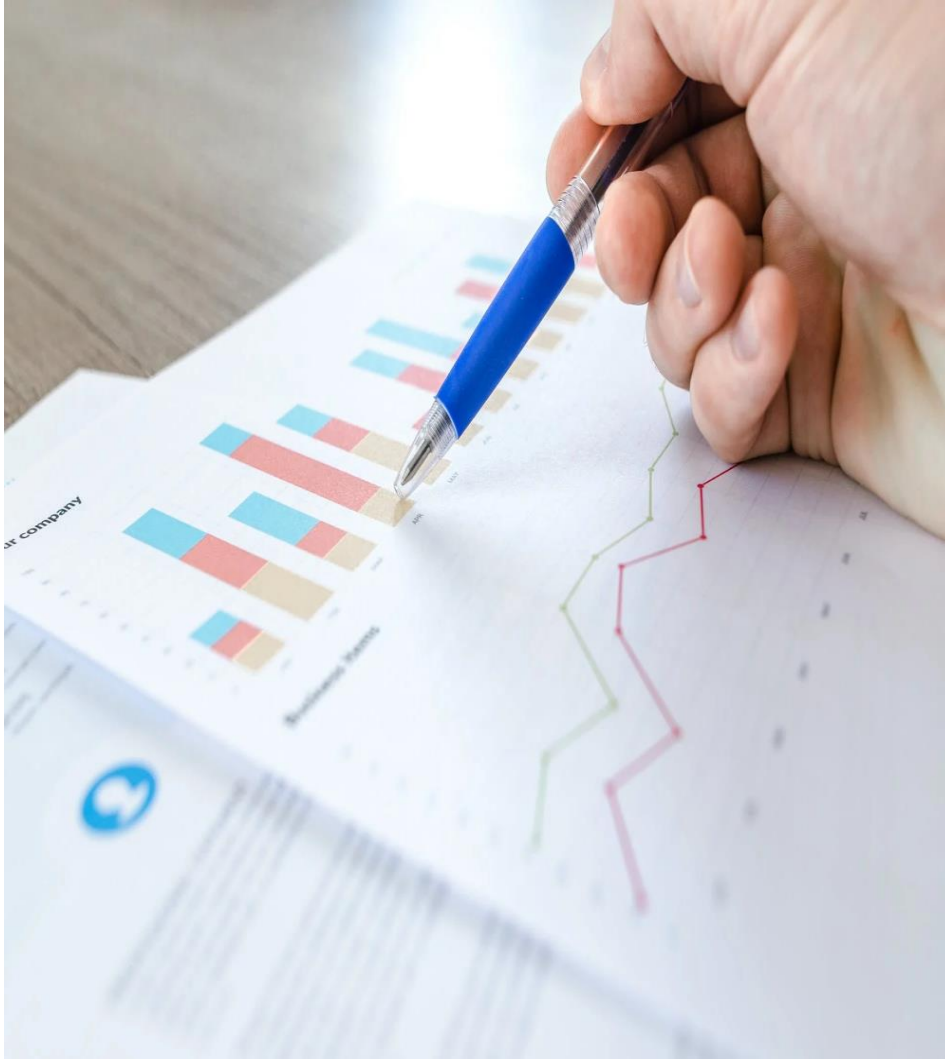


### Column-Based Storage Layout



### Hybrid-Based Storage Layout (row group size = 2)





## Delta Lake

- O Delta Lake é uma camada de armazenamento open-source que traz transações ACID (atomicidade, consistência, isolamento e durabilidade) para o Apache Spark e cargas de trabalho de Big Data.





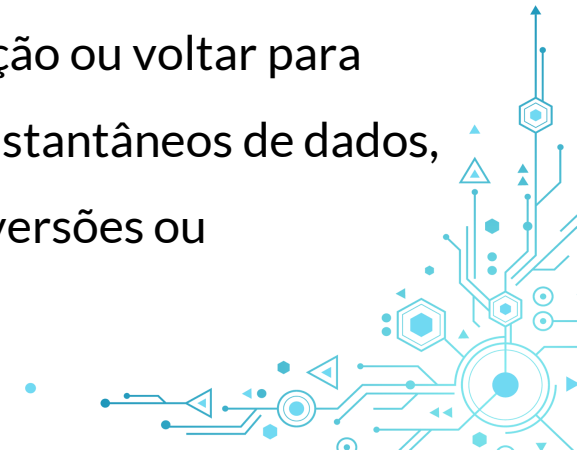
## Delta Lake

- Digamos que somos engenheiros de dados. Procuramos criar uma arquitetura de dados que facilite os Processos Analíticos Online ( OLAP ), que são apenas consultas seletivas voltadas para a análise de dados. Alguns exemplos de funções de dados otimizadas em um ambiente OLAP são análise exploratória de dados ou ciência da decisão.
- Velocidade de leitura : com que rapidez podemos acessar e decodificar informações relevantes de arquivos binários
- Tamanho no disco : quanto espaço nosso arquivo precisa quando em formato binário, compactação de 75% a 87% dependendo da versão que estiver usando

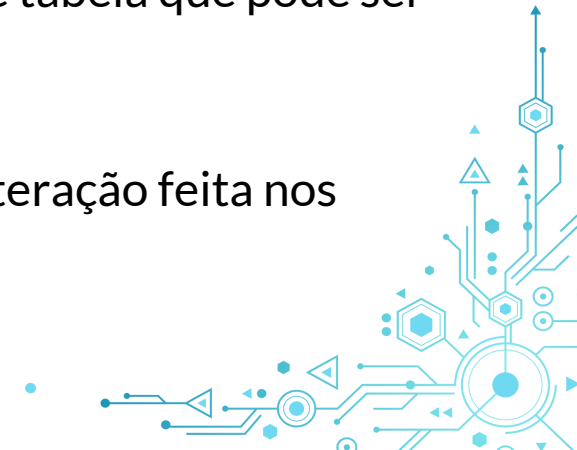


## Delta Lake

- **Transações ACID:** Os data lakes normalmente são preenchidos por vários processos e pipelines, alguns dos quais fazem a gravação de dados simultaneamente com as leituras. Antes do Delta Lake e da adição de transações, os engenheiros de dados tinham que passar por um processo manual propenso a erros para garantir a integridade dos dados. O Delta Lake traz as familiares transações ACID aos data lakes. Ele fornece capacidade de serialização, o nível mais alto de isolamento.
- **Manipulação de metadados escalonável:** No Big Data, até mesmo os metadados podem ser "Big Data". O Delta Lake trata metadados como dados, aproveitando o poder de processamento distribuído do Spark para lidar com todos os seus metadados. Como resultado, o Delta Lake pode lidar com tabelas de escala de petabytes com bilhões de partições e arquivos com facilidade.
- **Viagem no tempo (controle de versão de dados):** A capacidade de "desfazer" uma alteração ou voltar para uma versão anterior é um dos principais recursos das transações. O Delta Lake fornece instantâneos de dados, permitindo que você reverta para versões anteriores de dados para fins de auditorias, reversões ou reprodução de experimentos.



- **Formato Aberto:** Apache Parquet é o formato de linha de base do Delta Lake, permitindo que você aproveite os eficientes esquemas de compactação e codificação que são nativos do formato.
- **Lote Unificado e Fonte e Coletor de Streaming:** Uma tabela no Delta Lake é uma tabela de lote, bem como uma fonte e coletor de streaming. A ingestão de dados de streaming, o aterramento histórico de lote e as consultas interativas acabam funcionando imediatamente.
- **Imposição do esquema:** A imposição do esquema ajuda a garantir que os tipos de dados estejam corretos e que as colunas necessárias estejam presentes, impedindo que dados incorretos causem inconsistência de dados e também condições para atualizações, exclusões e mesclagem de dados dentro do esquema.
- **Evolução do esquema:** O Delta Lake permite que você faça alterações em um esquema de tabela que pode ser aplicado automaticamente, sem a necessidade de gravar uma DDL de migração.
- **Histórico de auditoria:** O log de transações do Delta Lake registra detalhes sobre cada alteração feita nos dados, fornecendo uma trilha de auditoria completa das alterações.



## Aplicações do Spark

- Para trabalharmos com o Spark, iremos utilizar o Python. A junção do Python + Spark deu origem ao PySpark, podemos pensar no PySpark como uma versão poderosa do Pandas para lidar com grande volumes de dados. Apesar de serem projetos diferentes, cabe a analogia.
- Todos os projetos que envolvem dados, o spark pode ser usado para extração, transformação e disponibilização, em qualquer área de negocio e tecnologia
  - API
  - Tabelas
  - Arquivos
  - Integrações



## Concorrentes e Ambientes

- O spark é framework de código aberto, podendo ser encontrado no git hub, gratuitamente e seu consumo e uso sem custo nenhum, porém exige que seu manuseador o conheça muito bem!
- Podemos rodar o spark em qualquer ambiente, ou seja, gerando muitos concorrente e ambientes para trabalharmos e construirmos soluções voltadas a dados.
- Local
  - K8s
  - On Premise
  - VM
  - Docker
  - Seu notebook (maquina local)
- Cloud
  - Amazon AWS
  - Microsoft Azure
  - Google Cloud
  - Digital Ocean
  - Databricks





**Databricks** é uma empresa fundada pelos criadores da Apache Spark que tem por objetivo ajudar clientes a processarem Big Data na nuvem utilizando Spark.

Foi fundada em 2013 por Ali Ghodsi, Andy Konwinski, Scott Shenker, Ion Stoica, Patrick Wendell, Reynold Xin, Matei Zaharia.



- O ambiente que será utilizado é o Databricks Community  
<https://community.cloud.databricks.com/>
- Esse é um ambiente comunitário, onde o Databricks disponibiliza gratuitamente um Cluster de **15GB de memória e 2 Cores** para realizar os experimentos.
- Cada conta, contém um cluster associado deverá ser utilizado durante o desenvolvimento.
  - Lembre-se que depois de 2 horas de inatividade, o mesmo será finalizado, sendo necessário criar um novo.



## Try Databricks

Launch cloud-optimized Apache Spark™ clusters in minutes

Tell us a little about yourself to get started.

* First Name:	* Last Name:
<input type="text" value="Dino"/>	<input type="text" value="Magri"/>
* Company Name	* Work Email
<input type="text" value="Aulas"/>	<input type="text" value="aulas.dinomagri@gmail.com"/>
* How would you describe your role?	* What is your intended use case?
<input type="text" value="Other"/>	<input type="text" value="Education - Learning &amp; Teaching"/>
Phone Number	
<input type="text"/>	

By Clicking "Sign Up", you agree to the [Privacy Policy](#)

☐ Keep me informed with occasional updates about Databricks and Apache Spark™.

SIGN UP



## Try Databricks

Launch cloud-optimized Apache Spark™ clusters in minutes

Select a platform

### DATABRICKS PLATFORM - FREE TRIAL

For businesses looking for a zero-management cloud platform built around Apache Spark

- Unlimited clusters that can scale to any size
- Job scheduler to execute jobs for production pipelines
- Fully interactive notebook with collaboration, dashboards, REST APIs
- Advanced security, role-based access controls, and audit logs
- Single Sign On support
- Integration with BI tools such as Tableau, Qlik, and Looker
- 14-day full feature trial (excludes cloud charges)

### COMMUNITY EDITION

For students and educational institutions just getting started with Apache Spark

- Single cluster limited to 6GB and no worker nodes
- Basic notebook without collaboration
- Limited to 3 max users
- Public environment to share your work

GET STARTED

By clicking "Get Started" for the Community Edition, you agree to the [Databricks Community Edition Terms of Service](#).

### GET STARTED ON



OR



Please note that Azure Databricks is provided by Microsoft and is subject to Microsoft's terms.

By clicking on the "AWS" button to get started, you agree to the [Databricks Terms of Service](#).



# Time to check your email!

Thank you for signing up. Now it's time to validate your email address.  
Please check the email you provided for next steps.



## Welcome to Databricks Community Edition!

Databricks Community Edition provides you with access to a free micro-cluster as well as a cluster manager and a notebook environment - ideal for developers, data scientists, data engineers and other IT professionals to get started with Spark.

We need you to verify your email address by clicking on [this link](#). You will then be redirected to Databricks Community Edition!

**Get started by visiting:** <https://community.cloud.databricks.com/login.html?resetpassword&username=aulas.dinomagri%40gmail.com&expiration=-60000&token=a90c992e0906b791bba18dc49e060ddfe8d9c92b>

If you have any questions, please contact [feedback@databricks.com](mailto:feedback@databricks.com).

- The Databricks Team





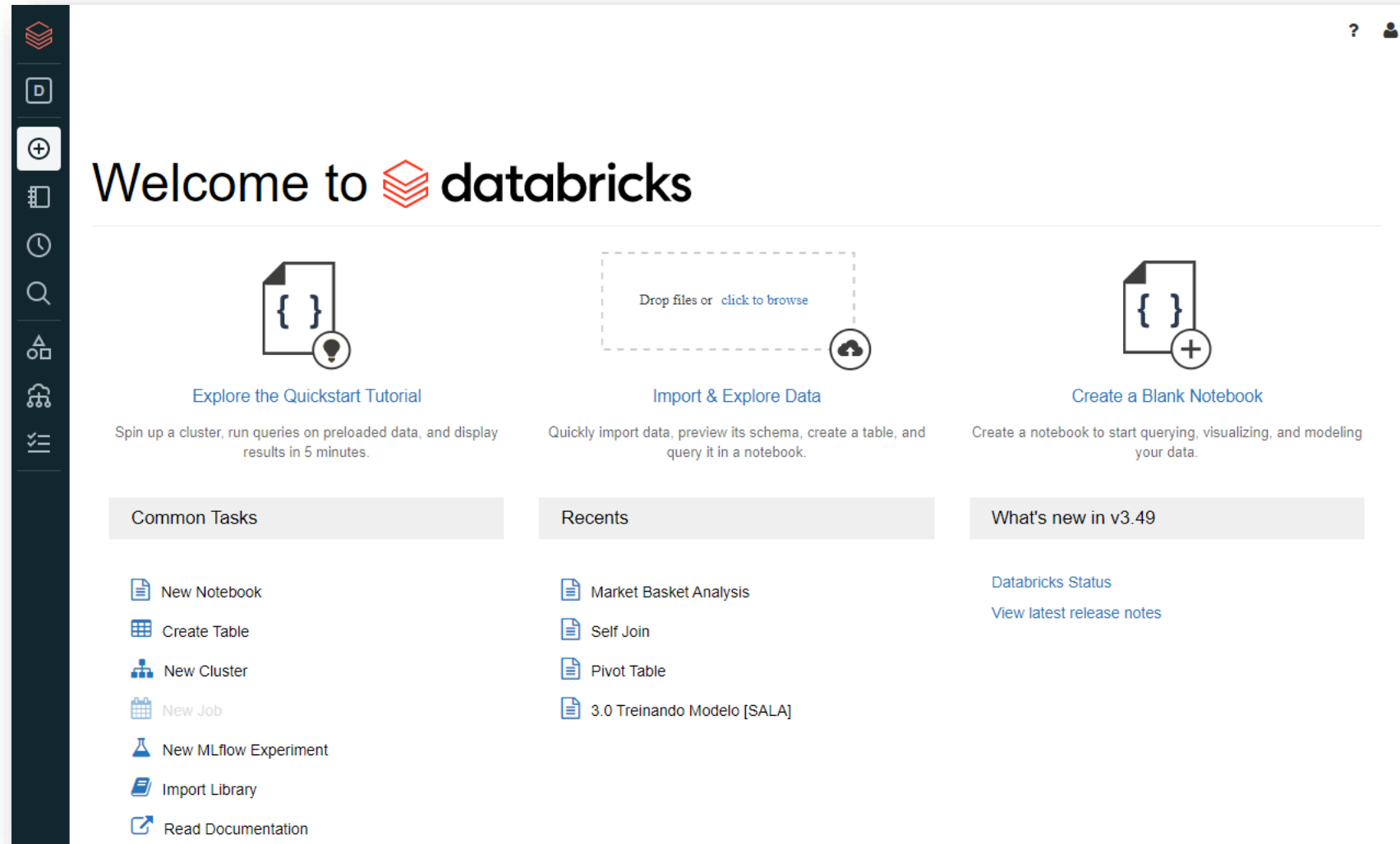
## Reset Password

Please enter your new password: \*

Please confirm your new password: \*

Reset password



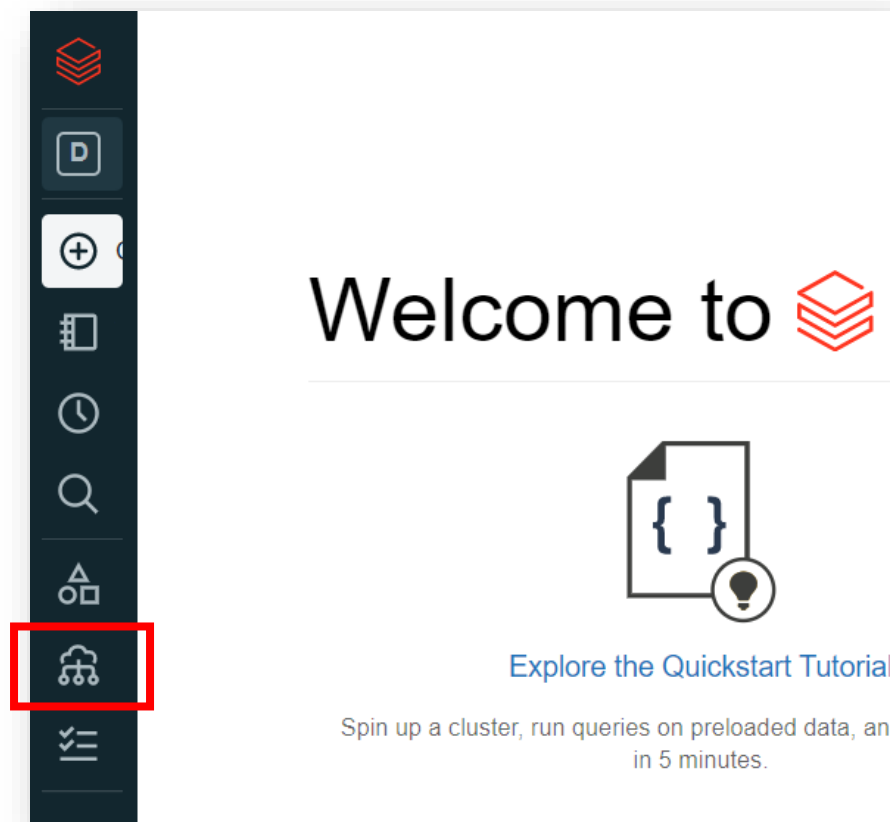


- A plataforma Databricks possibilita realizar boa parte do pipeline dos dados de maneira simples, rápida e eficaz.
- Para isso, precisamos entender os principais componentes e seus relacionamentos dentro dessa plataforma.
- Os principais componentes que iremos aprender são:
  - Workspace
  - Clusters
  - Notebooks
  - Tabelas
  - Jobs

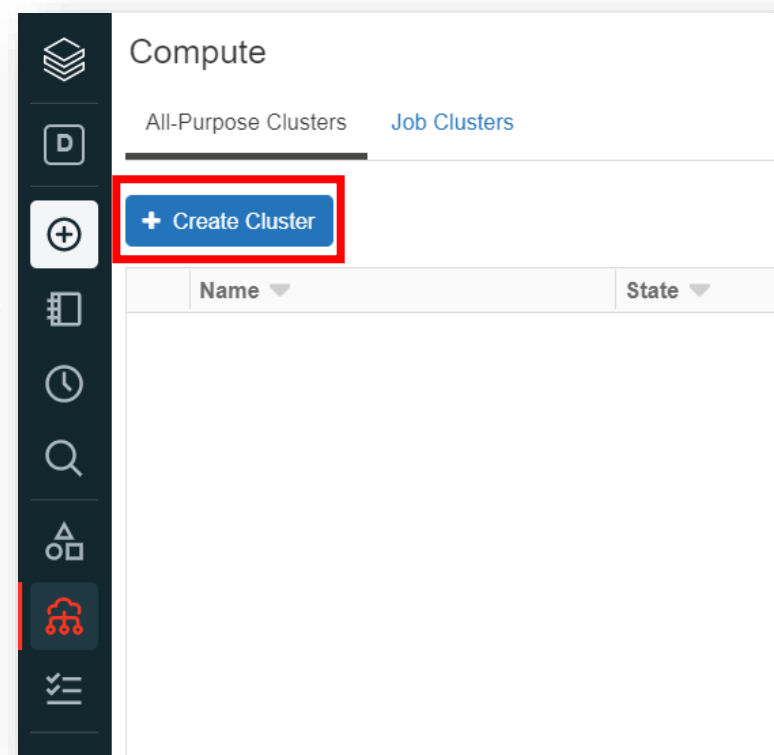


- Os clusters do Databricks fornecem uma plataforma de computação de cluster unificada para vários casos de uso.
- Podemos gerenciar os clusters utilizando a interface de usuário (UI), linha de comando (CLI) ou utilizando a API REST.





- Para rodarmos códigos temos que criar um cluster.
- No menu, clique em **Compute**.
- Na tela que irá abrir, clique em **+ Create Cluster**





The screenshot shows the 'Create Cluster' page in Databricks. A sidebar on the left contains navigation icons. The main content area is titled 'Create Cluster' and 'New Cluster'. It includes a 'Cancel' button and a 'Create Cluster' button, with annotation 3 pointing to the latter. Below these are fields for 'Cluster Name' (containing 'cluster-grupoX', with annotation 1 pointing to it), 'Databricks Runtime Version' (set to 'Runtime: 8.3 (Scala 2.12, Spark 3.1.1)', with annotation 2 pointing to it), and 'Availability Zone' (set to 'auto'). A note states: 'Databricks Runtime 8.x uses Delta Lake as the default table format. [Learn more](#)'. The 'Instance' section shows a warning: 'Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.' The 'Instances' tab is selected, showing 'Spark' as the instance profile. Resource specifications are listed at the top right: '0 Workers: 0 GB Memory, 0 Cores, 0 DBU' and '1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU'.

Create Cluster

New Cluster Cancel Create Cluster **0 Workers:** 0 GB Memory, 0 Cores, 0 DBU  
**1 Driver:** 15.3 GB Memory, 2 Cores, 1 DBU ?

Cluster Name  
cluster-grupoX

Databricks Runtime Version ?  
Runtime: 8.3 (Scala 2.12, Spark 3.1.1) | v

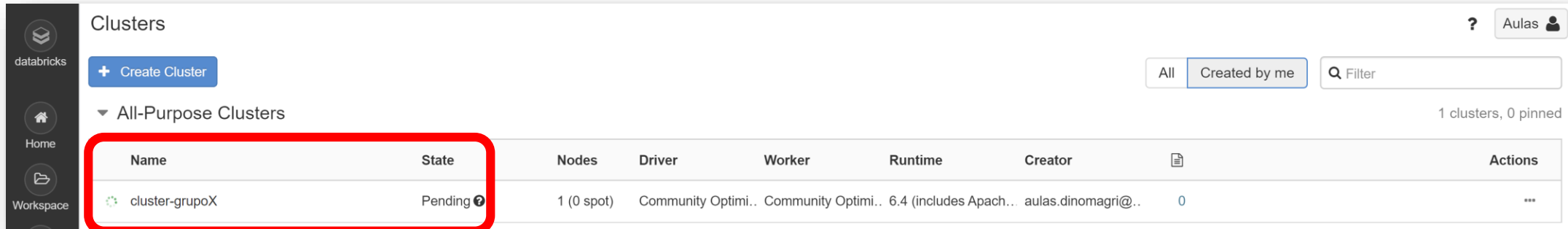
**Note** Databricks Runtime 8.x uses Delta Lake as the default table format. [Learn more](#)

Instance

Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

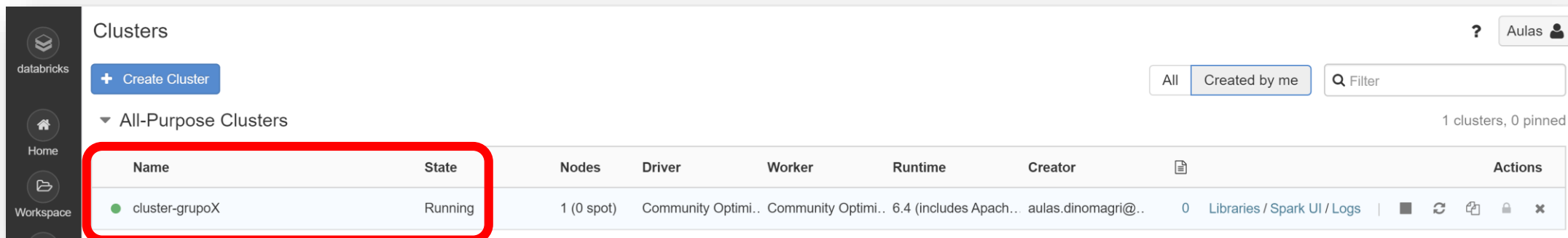
Instances **Spark**

Availability Zone ?  
auto | v



The screenshot shows the Databricks Clusters management interface. On the left sidebar, there are navigation icons for 'databricks', 'Home', and 'Workspace'. The main header 'Clusters' includes a '+ Create Cluster' button and a user profile 'Aulas'. Below the header, there's a filter section with 'All' and 'Created by me' buttons, and a search bar labeled 'Filter'. A status indicator shows '1 clusters, 0 pinned'. The main table lists clusters with columns: Name, State, Nodes, Driver, Worker, Runtime, Creator, and Actions. The first cluster, 'cluster-grupoX', is highlighted with a red box and has a 'Pending' state with a help icon.

Name	State	Nodes	Driver	Worker	Runtime	Creator	Actions
cluster-grupoX	Pending ?	1 (0 spot)	Community Optimi..	Community Optimi..	6.4 (includes Apach...	aulas.dinomagri@..	0



The screenshot shows the Databricks Clusters management interface, identical to the one above but with the cluster 'cluster-grupoX' now in a 'Running' state. The 'State' column now shows 'Running' with a green dot icon. The 'Actions' column now includes links for 'Libraries / Spark UI / Logs' and icons for refreshing, copying, and deleting the cluster.

Name	State	Nodes	Driver	Worker	Runtime	Creator	Actions
cluster-grupoX	Running	1 (0 spot)	Community Optimi..	Community Optimi..	6.4 (includes Apach...	aulas.dinomagri@..	0 Libraries / Spark UI / Logs   [refresh] [copy] [delete]

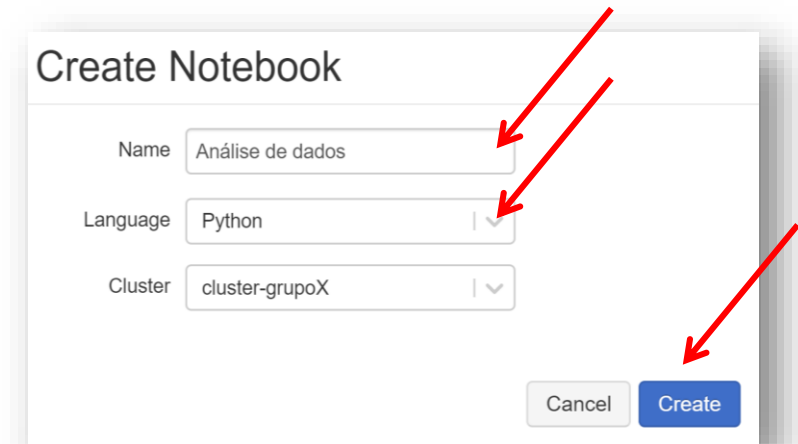
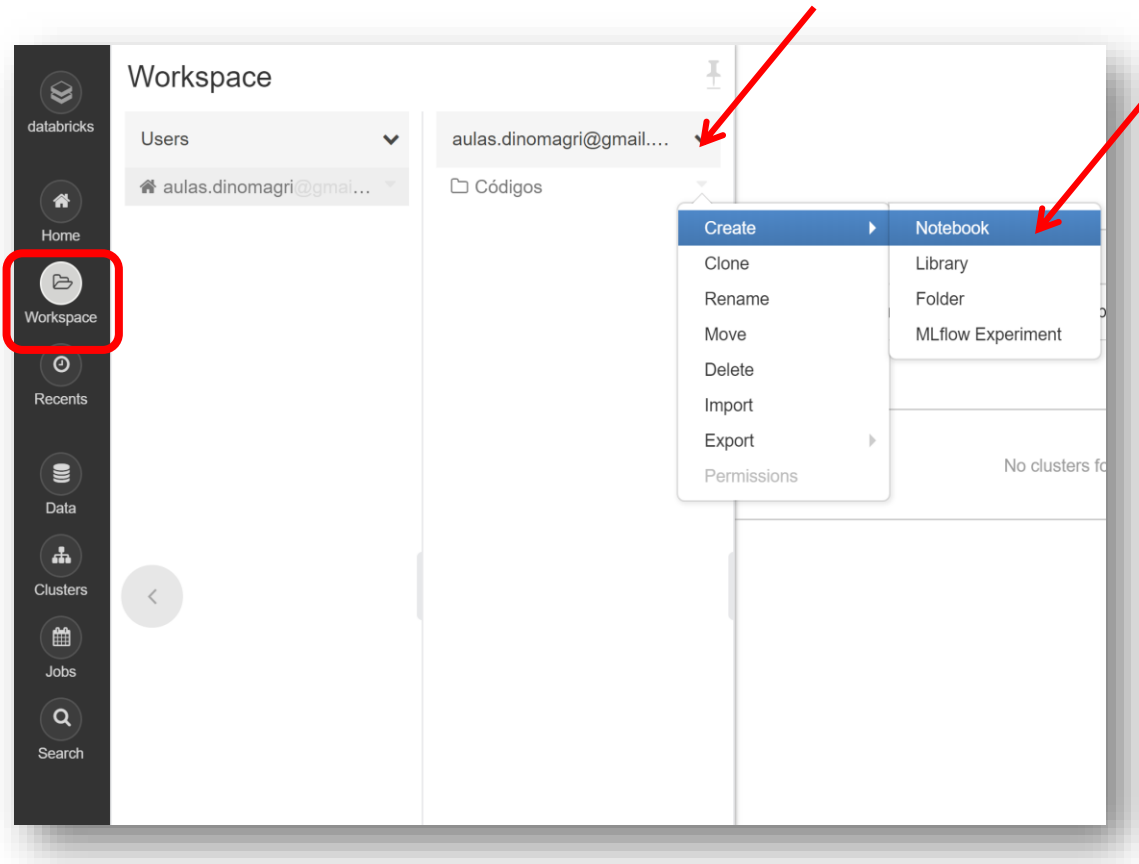


- A plataforma Databricks possibilita realizar boa parte do pipeline dos dados de maneira simples, rápida e eficaz.
- Para isso, precisamos entender os principais componentes e seus relacionamentos dentro dessa plataforma.
- Os principais componentes que iremos aprender são:
  - Workspace
  - Clusters
  - **Notebooks**
  - Tabelas
  - Jobs



- Como já vimos o Notebook é uma interface baseada na Web que contém código executável, visualização e texto narrativo.
- Todos organizados em células dentro dos notebooks. Ao usar um notebook, você está desenvolvendo e executando essas células principalmente à medida que itera os resultados.







The screenshot displays the Databricks workspace interface. On the left is a dark sidebar with navigation icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area is titled "Análise de dados (Python)" and shows a command execution for "cluster-grupoX".

**Cmd 1**

```
1 spark
```

**Out[1]:**

```
SparkSession - hive
SparkContext
Spark UI
Version
  v2.4.5
Master
  local[8]
AppName
  Databricks Shell
```

Command took 0.13 seconds -- by aulas.dinomagri@gmail.com at 15/04/2020 21:29:31 on cluster-grupoX

**Cmd 2**

```
1
```

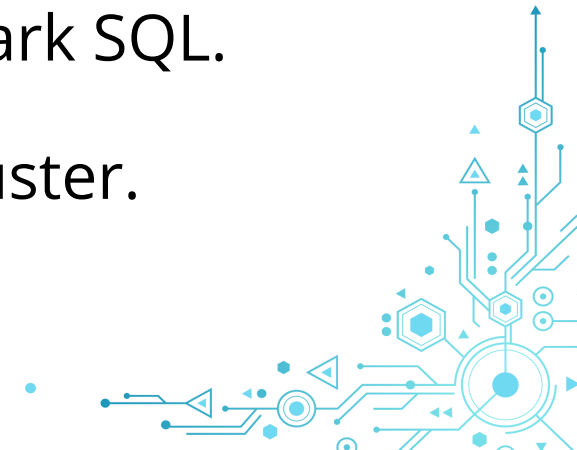
The top of the interface includes a toolbar with icons for File, View: Code, Permissions, Run All, Clear, Publish, Comments, Runs, and Revision history. A user profile icon labeled "Aulas" is in the top right corner.



- A plataforma Databricks possibilita realizar boa parte do pipeline dos dados de maneira simples, rápida e eficaz.
- Para isso, precisamos entender os principais componentes e seus relacionamentos dentro dessa plataforma.
- Os principais componentes que iremos aprender são:
  - Workspace
  - Clusters
  - Notebooks
  - **Tabelas**
  - Jobs



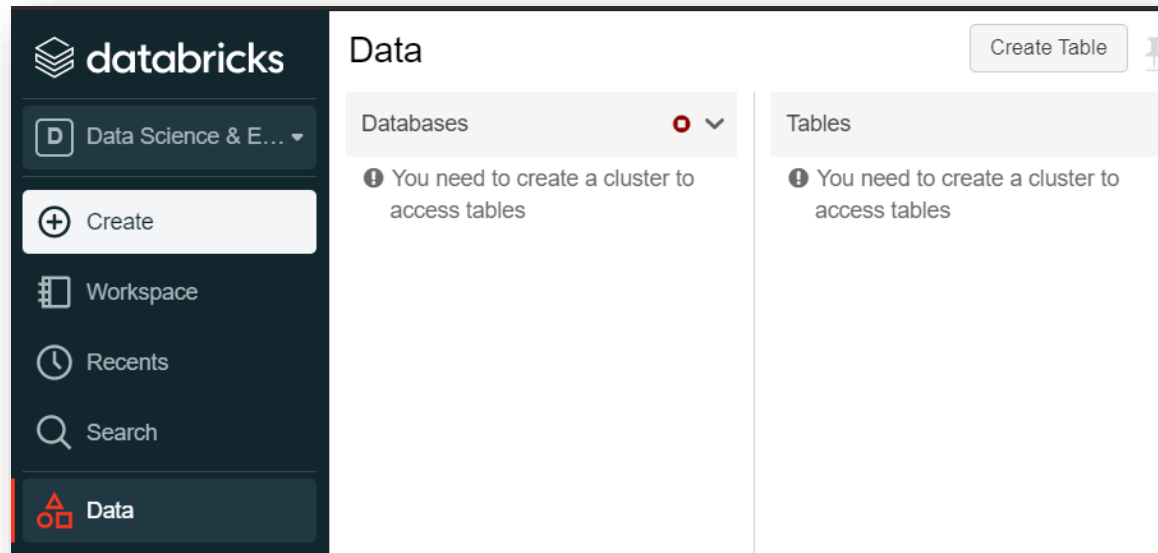
- Uma tabela no Databricks é uma coleção de dados estruturados.
- As tabelas são equivalentes aos DataFrames do Apache Spark.
- No Spark, um Dataframe é uma coleção distribuída de dados organizados em colunas nomeadas.
- Conceitualmente, é o equivalente a uma tabela em um banco de dados relacional ou a um Dataframe em R ou Python, mas é fornecido com otimizações muito mais ricas.
- Podemos consultar as tabelas via APIs Spark DataFrame e Spark SQL.
- Para podermos ver ou criar uma tabela, devemos criar um cluster.



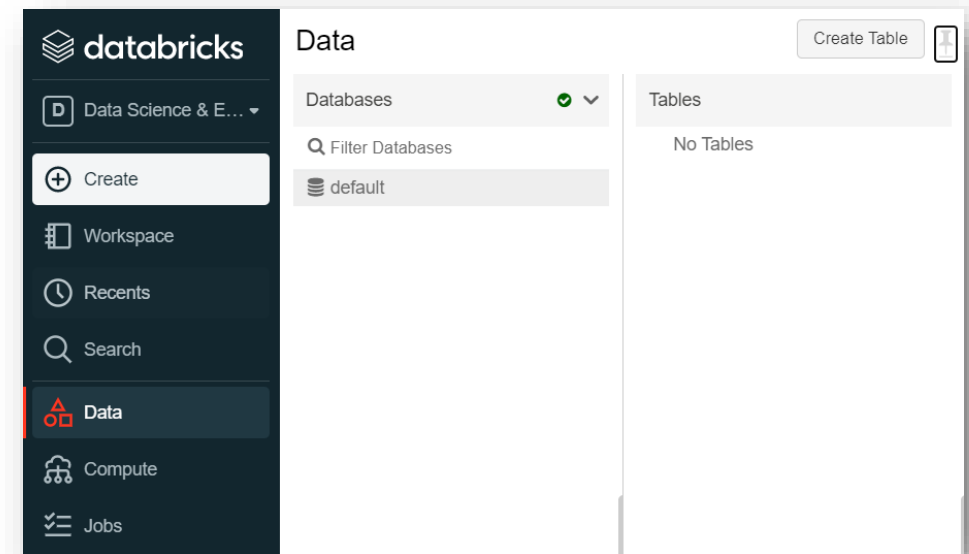
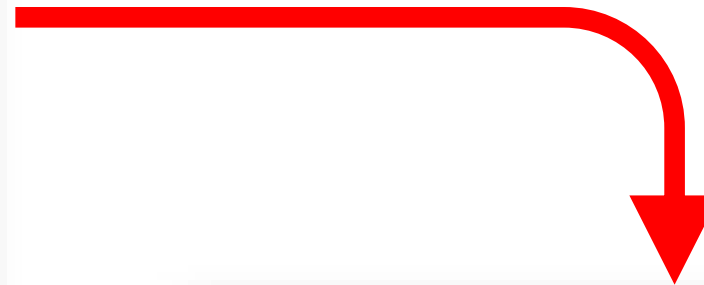
# AMBIENTE DATABRICKS - TABELAS

PROJETO FINAL

72



Depois de criado um cluster

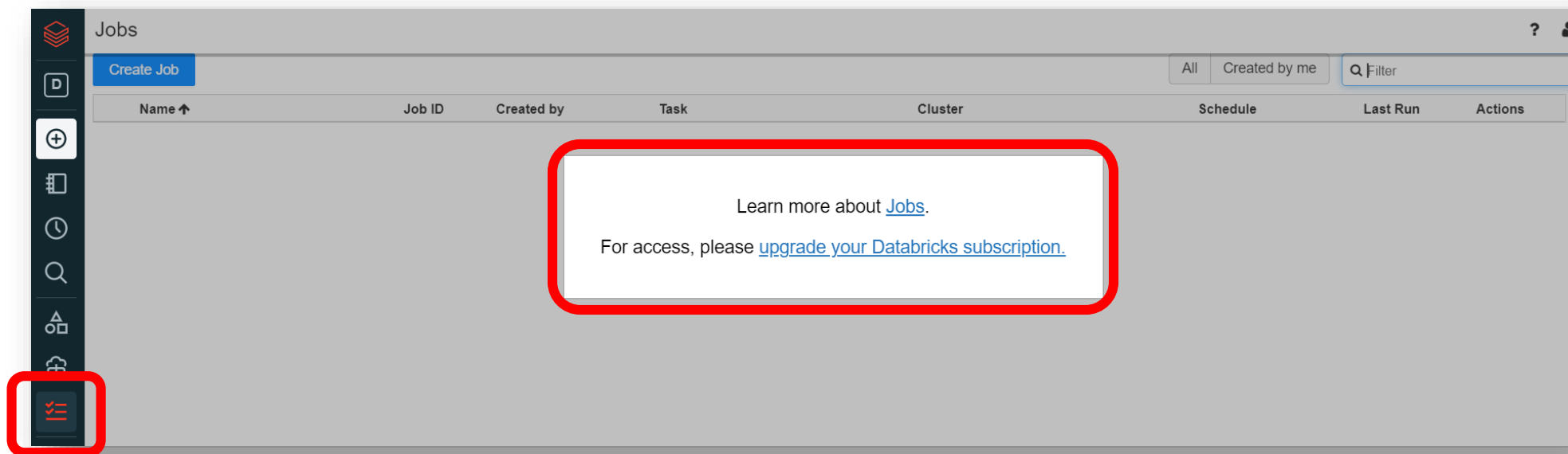


- A plataforma Databricks possibilita realizar boa parte do pipeline dos dados de maneira simples, rápida e eficaz.
- Para isso, precisamos entender os principais componentes e seus relacionamentos dentro dessa plataforma.
- Os principais componentes que iremos aprender são:
  - Workspace
  - Clusters
  - Notebooks
  - Tabelas
  - **Jobs**



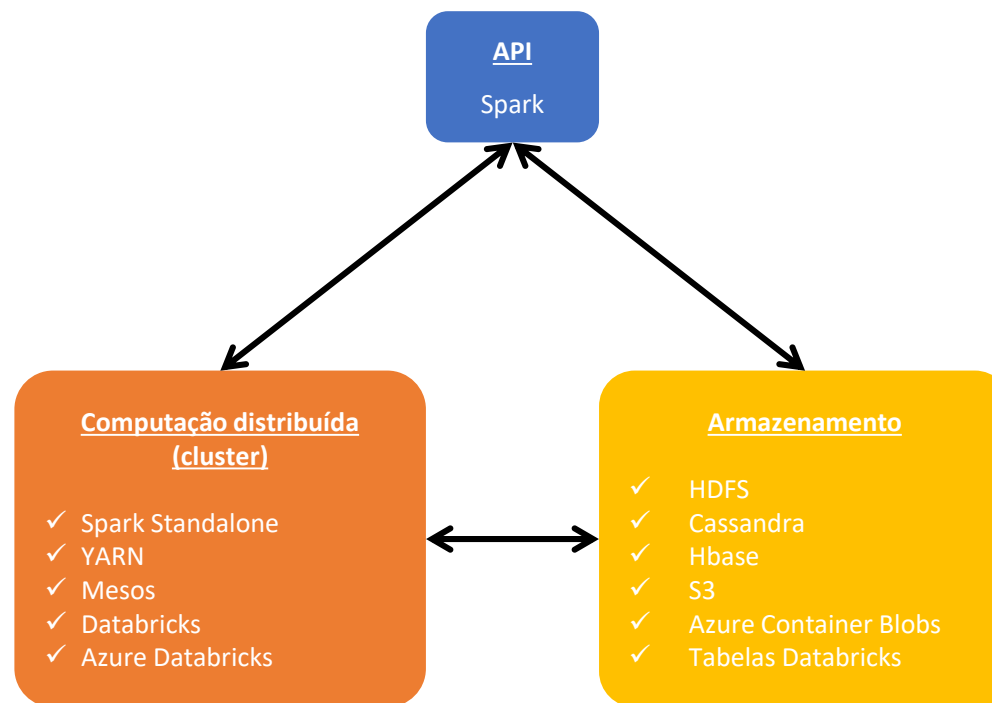


- Podemos agendar **jobs** específicos para executar tarefas repetitivas, por exemplo, realizar uma ingestão diária de dados em um pipeline ETL.
- Uma coisa importante a salientar é que a funcionalidade de *jobs* não está disponível para o Databricks Community.



## Como as aplicações Spark são construídas?

O processamento distribuído é realizado pela API do Spark, que se comunica com a **computação** e o **armazenamento** necessário.



- O Spark suporta 4 tipos de linguagem de programação para acessar suas funcionalidades:
  - **Python** 😊
  - Scala
  - Java
  - R
- Algumas funcionalidades (principalmente as mais novas) podem estar disponível somente em Scala, que é a linguagem utilizada no desenvolvimento do Spark.



- Para acessar a API do Spark é necessário que o programa que iremos criar, tenha acesso as funcionalidades via contexto do Spark (sparkContext ou sparkSession).
- Como vimos podemos utilizar diversas linguagens de programação para acessar o Spark e iremos utilizar o Python para criar nossas aplicações.
- Nesse ambiente iremos utilizar o Spark no modo interativo com o uso dos notebooks.



## HANDS-ON





## REFERÊNCIAS BIBLIOGRÁFICAS



- **Spark: The Definitive Guide: Big Data Processing Made Simple** - Bill Chambers (Author), Matei Zaharia (Author) - USA: O'Reilly, 2018
- **Advanced Analytics with Spark** - Sandy Ryza, Uri Laserson, Sean Owen, and Josh Wills – USA: O'Reilly, 2015
- **Apache Spark 2.x Cookbook** – Rishi Yadav, Packt Publishing, 2017.



- **Mastering Apache Spark 2.x - Second Edition** - Romeo Kienzler, 2017.
- **Learning Spark 2** – Muhammad Asif Abbasi, Packt Publishing, 2017.
- **Spark 2.0 for Beginners** – Rajanarayanan Thottuvaikkatumana – Packt Publishing, 2017.
- **Spark for Python Developers** – Amit Nandi, Packt Publishing, 2015.



- **Spark for Python Developers** – Amit Nandi, Packt Publishing, 2015.
- **Fast Data Processing with Spark** – Krishna Sankar, Holden Karau – Packt Publishing, 2015.
- **Spark for Python Developers** – Amit Nandi, Packt Publishing, 2015.
- **Apache Spark 2.x for Java Developers** – Sourav Gulati and Sumit Kumar, Packt Publishing, 2017.



- **Python for kids – A playful Introduction to programming** – Jason R. Briggs – San Francisco – CA: No Starch Press, 2013.
- **Python for Data Analysis** – Wes McKinney – USA: O'Reilly, 2013.
- **Python Cookbook** – David Beazley & Brian K. Jones – O'Reilly, 3th Edition, 2013.

