

# TEMARIO

jueves, 12 de diciembre de 2024 9:58

# CALENDAR

Junes, 28 de octubre de 2024 18:12

1Q	2410-22_2MaBDBA_FT-EN - Máster in Big Data & Business Analytics		PV 204
	2410-22_2MaBDBA_FT-EN-02A - Big Data Processing	Príncipe de Vergara 156	
	Martinez Arribas, Jose Luis	Estado	
28-10-24	11:30-13:30		
30-10-24	11:30-13:30		
6-11-24	11:30-13:30		
7-11-24	11:30-13:30		
11-11-24	11:30-13:30		
13-11-24	11:30-13:30		
18-11-24	11:30-13:30		
20-11-24	11:30-13:30		
26-11-24	11:30-13:30		
2-12-24	11:30-13:30		
4-12-24	11:30-13:30		
10-12-24	11:30-13:30		
12-12-24	11:30-13:30		
16-12-24	11:30-13:30		
18-12-24	11:30-13:30		
8-1-25	11:30-13:30		
13-1-25	11:30-13:30	High level programming languages: Spark	
15-1-25	11:30-13:30	ETL process: Kafka+python+postgres	
20-1-25	11:30-13:30	Data Workflows: Apache Airflow (batch)	
22-1-25	09:00-11:00	Data Processing--- QUALITY OF DATA	
22-1-25	11:30-13:30	Data Processing--- QUALITY OF DATA	
27-1-25	11:30-13:30	Designing for Scalability+ deployment apps in Cloud - Data Modeling for Business Intelligence	
5-2-25	11:30-13:30	Recap of All - Architectures for Distributed Systems-Real use case SmartHomes	
10-2-25	11:30-13:30	Exam	

# CONTACT

miércoles, 30 de octubre de 2024 11:50

+34639002090

[jlmartinez@ensaco.es](mailto:jlmartinez@ensaco.es)

[luis.martinez@campuseae.es](mailto:luis.martinez@campuseae.es)

WIFI

lunes, 28 de octubre de 2024 18:24

EAE  
EAEbspv156

# Glossary

lunes, 28 de octubre de 2024 22:55

Data Life Cycle:

ERP:

CRM: costumer relationship manager

IDE: integrated development environment

Powerquery (data)

Powerpivot

Power automate

ETL: Extract, Transform and Load

RPA: Robotic Process Automation

GUI: Graphical User Interface

Windows+V= Clipboard

Webhook = to automate communication between application programming interfaces (APIs). It can be used to activate workflows.

# WHAT IS?

jueves, 12 de diciembre de 2024 9:55

# Visual Studio Code

miércoles, 30 de octubre de 2024 7:10

[My Python Development Environment Setup - Full Tutorial](#)



Sure! Here is a markdown file with a comprehensive guide to learning Visual Studio Code:

## Guide to Learning Visual Studio Code

Visual Studio Code (VS Code) is a powerful and popular code editor that provides a wide range of features to support development across multiple languages and frameworks. This guide will help you get started and make the most out of this fantastic tool.

### Table of Contents

1. [Installation](#)
2. [User Interface Overview](#)
3. [Basic Setup](#)
4. [Extensions](#)
5. [Customization](#)
6. [Keyboard Shortcuts](#)
7. [Working with Projects](#)
8. [Debugging](#)
9. [Version Control with Git](#)
10. [Tips and Tricks](#)
11. [Resources for Further Learning](#)

### Installation

1. **Download:** Visit [Visual Studio Code's website](#) and download the latest version for your operating system.
2. **Installation:** Follow the instructions for your platform (Windows, macOS, or Linux) to install the editor.
3. **Launch:** Open VS Code after installation.

## User Interface Overview

The main components of the VS Code user interface are:

1. **Activity Bar:** Located on the far left, it contains icons for key features like Explorer, Search, Source Control, Debug, and Extensions.
2. **Side Bar:** Displays views like Explorer or Source Control based on the selection in the Activity Bar.
3. **Editor:** The main area where you edit your code.
4. **Panel:** Displays output, integrated terminal, and other information.
5. **Status Bar:** Shows useful information about your workspace, like file encoding, Git branch, and language mode.

## Basic Setup

1. **Theme and Appearance:**
  - Go to File > Preferences > Color Theme to change the theme.
  - Use File > Preferences > Settings to customize font size, editor layout, and more.
2. **Settings Sync:** If you work across multiple devices, enable Settings Sync under the Account menu to sync your settings, themes, extensions, and keybindings.

## Extensions

Extensions are crucial to enhancing your productivity in VS Code. Here are some must-have extensions:

1. **Language Support:** Add extensions for languages you work with, like Python, JavaScript, C#, etc.
2. **Prettier:** A code formatter for consistent and clean code.
3. **ESLint:** For linting JavaScript/TypeScript code.
4. **Live Server:** Quickly launch a development server with live reloading for web development.
5. **GitLens:** Provides advanced Git capabilities.

To install extensions:

1. Open the Extensions view by clicking on the Extensions icon in the Activity Bar or pressing **Ctrl + Shift + X**.
2. Search for an extension and click **Install**.

## Customization

1. **Settings:**
  - Open File > Preferences > Settings or use **Ctrl + ,**.
  - Use the search bar to find specific settings.
2. **Keybindings:**
  - Customize shortcuts under File > Preferences > Keyboard Shortcuts or press **Ctrl + K, Ctrl + S**.
  - Search for commands and change keybindings as needed.
3. **User Snippets:**
  - Create code snippets for frequently used code blocks.
  - Go to File > Preferences > User Snippets and select a language to create snippets for.

## Keyboard Shortcuts

- **Command Palette:** **Ctrl + Shift + P** (Windows/Linux) or **Cmd + Shift + P** (macOS)
- **Quick Open:** **Ctrl + P** (Windows/Linux) or **Cmd + P** (macOS)
- **Toggle Terminal:** **Ctrl + `** (backtick) (Windows/Linux) or **Cmd + `** (macOS)
- **Split Editor:** **Ctrl + \** (Windows/Linux) or **Cmd + \** (macOS)
- **Format Document:** **Shift + Alt + F** (Windows/Linux) or **Shift + Option + F** (macOS)
- **Comment Line:** **Ctrl + /** (Windows/Linux) or **Cmd + /** (macOS)

## Working with Projects

1. **Opening a Folder:** Use **File > Open Folder** to open a project folder.
2. **Workspace Management:** Save workspaces for projects with

multiple folders.

3. **File Navigation:** Use Ctrl + P to quickly open files by name.
4. **Search:** Use the Search feature in the Side Bar or press Ctrl + Shift + F to search across your project.

## Debugging

1. **Setup Debug Configuration:**
  - Click on the Debug icon in the Activity Bar.
  - Create a launch configuration file for your project if needed.
2. **Debugging Features:**
  - Set breakpoints by clicking in the gutter next to a line number.
  - Use the Debug toolbar to start, pause, and step through your code.
  - View variables, call stack, and output in the Debug view.

## Version Control with Git

1. **Initialization:** Initialize a Git repository for your project or clone an existing one.
2. **Stage and Commit:**
  - Use the Source Control view to stage changes and commit them.
  - Use git status in the integrated terminal to view changes.
3. **Branching:** Manage branches and merge changes using the built-in Git features.
4. **Pushing and Pulling:** Use the Source Control view or terminal commands to push and pull code from remote repositories.

## Tips and Tricks

1. **Multi-cursor Editing:** Hold Alt (Windows/Linux) or Option (macOS) and click to place multiple cursors.
2. **Emmet Abbreviations:** Use Emmet for faster HTML and CSS writing.
3. **Zen Mode:** Press Ctrl + K Z for a full-screen, distraction-free coding experience.
4. **Integrated Terminal:** Open a terminal with Ctrl + and use it to run scripts, Git commands, etc.

## Resources for Further Learning

1. **Official Documentation:** [VS Code Docs](#)
2. **YouTube Tutorials:** Search for beginner to advanced tutorials on YouTube.
3. **GitHub Repositories:** Explore open-source projects to learn best practices.
4. **Online Courses:** Platforms like Udemy, Coursera, and Pluralsight have courses dedicated to mastering VS Code.

Enjoy coding and exploring the full potential of Visual Studio Code!

You can copy this markdown content and use it in your guide! Let me know if you need any changes or additions.

# Markdown

domingo, 3 de noviembre de 2024 8:19

<https://github.com/jamesqquick/markdown-worksheet?tab=readme-ov-file>

<https://github.com/jamesqquick/markdown-worksheet/blob/master/worksheet.md>

# Guía para Aprender Visual Studio Code con Markdown

## Introducción a Visual Studio Code

Visual Studio Code (VSCode) es un editor de código fuente gratuito y potente, ideal para trabajar con Markdown. Ofrece soporte integrado para Markdown, permitiendo una edición y previsualización eficientes.

## Instalación de Visual Studio Code

1. Descarga VSCode desde su sitio web oficial.
2. Instala el programa siguiendo las instrucciones del instalador.

## Configuración Básica

### Abrir un Archivo Markdown

- Abre VSCode y selecciona `File > Open File...` para abrir tu archivo `.md`.

### Previsualización de Markdown

- Abre la previsualización con `Ctrl+Shift+V` o `Ctrl+K V` para ver los cambios en tiempo real.

## Extensiones Recomendadas

- \*\*Markdown All in One\*\*: Mejora las capacidades de edición y previsualización.
- \*\*markdownlint\*\*: Ayuda a mantener la sintaxis correcta.
- \*\*Code Spell Checker\*\*: Verifica la ortografía en tus documentos.

## Sintaxis Básica de Markdown

### Encabezados

```
```markdown
# Título Principal
## Subtítulo
### Subtítulo de Nivel 3
````
```

### Listas

- Elemento 1
- Elemento 2

1. Elemento numerado 1
2. Elemento numerado 2

### ### Énfasis

- \*Cursiva\*: `\*texto\*` o `'\_texto\_`
- \*\*Negrita\*\*: `\*\*texto\*\*` o `\_\_texto\_\_`
- \*\*\*Combinado\*\*\*: `\*\*\*texto\*\*\*`

### ### Bloques de Código

```
```markdown
```python
print("Hola, Mundo")
```
```

```

### ### Tablas

```
```markdown
| Columna 1 | Columna 2 |
|-----|-----|
| Dato 1   | Dato 2   |
```

```

## ## Personalización de VSCode para Markdown

- Accede a `Settings` para ajustar la fuente y el tamaño de la previsualización.
- Usa atajos como `Ctrl+K Ctrl+S` para personalizar comandos.

Con esta guía, podrás comenzar a trabajar eficazmente con Markdown en Visual Studio Code, aprovechando sus herramientas y extensiones para mejorar tu flujo de trabajo.

### Fuentes

- [1] VSCode Markdown Basico - Lemoncode TV <https://www.lemoncode.tv/curso/vscode-markdown/leccion/basico>
- [2] Sintaxis Markdown al completo - Cheatsheet en español <https://markdown.es/sintaxis-markdown/>
- [3] How to View Markdown in Visual Studio Code - MarkdownGenius  
<https://blog.mdconvt.com/how-to-view-markdown-in-visual-studio-code/>
- [4] El lenguaje Markdown - NetMentor <https://www.netmentor.es/entrada/lenguaje-markdown>
- [5] Paquete de creación de Learn para Visual Studio Code <https://learn.microsoft.com/es-es/contribute/content/how-to-write-docs-auth-pack>
- [6] Markdown Extension - Visual Studio Code <https://code.visualstudio.com/api/extension-guides/markdown-extension>
- [7] How to Use Markdown in VSCode – Syntax and Examples  
<https://www.freecodecamp.org/news/how-to-use-markdown-in-vscode/>
- [8] Markdown editing with Visual Studio Code  
<https://code.visualstudio.com/Docs/languages/markdown>

# SQL and NOSQL

domingo, 3 de noviembre de 2024

9:10

SQL:

PostgreSQL

[PostgreSQL in 100 Seconds](#)



[Learn PostgreSQL Tutorial - Full Course for Beginners](#)



NOSQL:

\*\*Mongo\*\*

[Complete MongoDB Tutorial](#)



[MySQL vs MongoDB](#)



# HADOOP

miércoles, 4 de diciembre de 2024 13:32

[What Is Hadoop? | Introduction To Hadoop | Hadoop Tutorial For Beginners | Simplilearn](#)



### **Prompting:**

1. Begin with an action verb: Generate, analyze, resume
2. Add always context to the task:
3. Introduce clear examples that guides to the response you want
4. Define the person that aligns with the task
5. Indicate the format of the output you would like ChatGPT to answer
6. You need to indicate the tone. Professional, Friendly, etc.
7. Document your prompts: create new things and add changes from time to time.

You are a prompt engineer.

Generate a list of steps that will add quality and good outcome results to your prompts.

Here you have an output of what I'm looking for:

### **"Prompting:**

- 1 Begin with an action verb: Generate, analyze, resume
- 2 Add always context to the task:
- 3 Introduce clear examples that guides to the response you want
- 4 Define the person that aligns with the task
- 5 Indicate the format of the output you would like ChatGPT to answer
- 6 You need to indicate the tone. Professional, Friendly, etc.
- 7 Document your prompts: create new things and add changes from time to time.

The target of this guide is for people that like to improve themselves and use ChatGPT to get more knowledge and better results for both work and like.

## **Prompt Engineering Guide for Optimal Results**

### **1. Start with a Clear Action Word**

- Use verbs that directly convey what the assistant should do, like: "Explain," "List," "Summarize," or "Generate."

### **2. Provide Relevant Context**

- Specify any background information that might influence the response, such as industry, audience, or specific knowledge area.

### **3. Give Detailed Instructions**

- Be explicit about the requirements, like "in 3-5 sentences" or

“include bullet points.”

#### **4. Include Examples**

- Offer sample responses or illustrations that showcase your desired outcome to improve clarity.

#### **5. Define the Audience**

- Specify the type of person or professional perspective, like “Explain for a beginner,” or “Answer as a project manager.”

#### **6. State the Tone and Style**

- Direct the assistant on how to sound: “Use a formal tone,” “Be concise and technical,” or “Answer in a friendly, conversational style.”

#### **7. Specify the Format**

- Request a particular structure, like “Create a table,” “Answer in steps,” or “Present as a list of pros and cons.”

#### **8. Experiment and Iterate**

- Document your prompts and refine them based on previous results, noting what works best for different tasks.

#### **9. Avoid Ambiguity**

- Minimize vague terms and make requests specific, avoiding open-ended or overly broad questions.

#### **10. Set Length Expectations**

- If you need concise answers or in-depth explanations, specify word counts or level of detail (e.g., “Give a brief summary in 100 words”).

#### **11. Encourage Creative Input**

- For creative tasks, invite ideas with a prompt like, “Suggest new approaches,” or “Offer innovative examples.”

#### **12. Use Constraints for Consistency**

- Limit responses with boundaries like “only use simple language” or “keep within two paragraphs.”

This guide empowers those who seek self-improvement and knowledge expansion, using prompt techniques to achieve high-quality and tailored responses from ChatGPT for both professional and personal insights.

# REFERENCIAS

jueves, 12 de diciembre de 2024 9:53

# Webs

Monday, October 23, 2023 10:06 AM

Kaggle.com

# AUTOHOTKEY

jueves, 12 de mayo de 2022 8:49

## Automatizador de tareas en Windows

Ideal para implementar en algún proceso repetitivo que viene bien lanzarlo con una atajo de teclado o una cadena de texto personalizada.

Menús, mover el ratón

Pará qué sirve?

1. Download Autohotkey in the website:
2. Install it
3. Ahk script --> file .ahk and run it

Manual: <https://www.autohotkey.com/docs/AutoHotkey.htm>

Recursos para Aprenderlo:

Referentes (Youtubers)

1. Joe Glines (<https://www.the-automator.com/>) --> este es el Master of the Universe
2. Tab Nation
3. Juho Lee
4. Anders Jensen (poco AHK, pero toca todo tema de RPAs y es interesante).

```
#NoEnv ; Recommended for performance and compatibility with future AutoHotkey releases.  
; #Warn ; Enable warnings to assist with detecting common errors.  
SendMode Input ; Recommended for new scripts due to its superior speed and reliability.  
SetWorkingDir %A_ScriptDir% ; Ensures a consistent starting directory.
```

```
:*:C:ME::  
Send Mi Empresa S.L  
return
```

```
:*:mesl::  
Send Mi Empresa S.L`nC/ Lirios, 9 entresuelo derecha`n50015 Zaragoza`nB9900000  
return
```

```
:C:MK::  
Send Marie Kondo  
return
```

```
:*:jl@::  
Send jlmiralles@toc.es  
return
```

```
::ma@::  
Send mariekondo@toc.es  
return
```

```
:*:sph::  
(  
Showroom Smart Passivhaus
```

Paseo de Europa 12, local 1  
28703, San Sebastián de los Reyes, Madrid, España  
Teléfono: 91XXXXXX  
)

::bd::  
Send, Buenos días ,`n`n{Left 3}  
return

::usl::  
Send, Un saludo,`n`n  
return

::;;::  
Send /;;;;;;;/;;;;;;;  
return

^SPACE:: Winset, Alwaysontop, , A

::]ip::  
IP:= GetPublicIP()  
Send %IP%  
;MsgBox % GetPublicIP() ;Display the networks public IP  
;MsgBox % GetLocalIPByAdaptor("Ethernet") ;Display the local IP of the adaptor named "Ethernet"  
return

For AdaptorName, IP in GetLocalIPs() { ;Display the IPs of all adaptors that have one  
    MsgBox, %AdaptorName%: %IP%  
}

GetPublicIP() {  
    HttpObj := ComObjCreate("WinHttp.WinHttpRequest.5.1")  
    HttpObj.Open("GET","https://www.google.com/search?q=what+is+my+ip&num=1")  
    HttpObj.Send()  
    RegexMatch(HttpObj.ResponseText,"Client IP address: ([\d\.]+)",match)  
    Return match1  
}

GetLocalIPByAdaptor(adaptorName) {  
    objWMIService := ComObjGet("winmgmts:{impersonationLevel = impersonate}!\\.\root\cimv2")  
    colItems := objWMIService.ExecQuery("SELECT \* FROM Win32\_NetworkAdapter WHERE  
NetConnectionID = "" adaptorName """.\_NewEnum, colItems[objItem]  
    colItems := objWMIService.ExecQuery("SELECT \* FROM Win32\_NetworkAdapterConfiguration  
WHERE InterfaceIndex = "" objItem.InterfaceIndex """.\_NewEnum, colItems[objItem]  
    Return objItem.IPAddress[0]  
}

GetLocalIPs() {  
    adapters := Object()  
    ips := Object()  
    objWMIService := ComObjGet("winmgmts:{impersonationLevel = impersonate}!\\.\root\cimv2")  
    colItems := objWMIService.ExecQuery("SELECT \* FROM Win32\_NetworkAdapter").\_NewEnum,  
    colItems[objItem]  
    While (colItems[objItem])  
        adapters.Insert(objItem.InterfaceIndex,objItem.NetConnectionID)

```
For index, name in adaptors {
    colItems := objWMIService.ExecQuery("SELECT * FROM Win32_NetworkAdapterConfiguration
WHERE InterfaceIndex = "" index """._NewEnum, colItems[objItem]
    If (name && objItem.IPAddress[0])
        ips.Insert(name,objItem.IPAddress[0])
    }
    Return ips
}
```

```
::]dd::
FormatTime, TimeString, A_Now, yyMMdd
Send 28%TimeString%1-SH
return
```

# CHOCOLATEY - Instalar fácil apps en Windows

martes, 10 de mayo de 2022 8:55

Instalación rápida de aplicaciones en Windows – Herramienta Chocolatey

Url de referencia: <https://chocolatey.org/>

1. Instalar la herramienta chocolatey a través del PowerShell (permisos de administrador)

```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

2. Instalar las herramientas a elección.

- a. choco install **powerbi**
- b. choco install **r.project**
- c. choco install **r.studio**
- d. choco install **qgis**
- e. choco install **python3**
- f. choco install **pycharm-community**
- g. choco install mysql --params "/port:3307 /serviceName:**MasterBD**"

# Scripts

miércoles, 25 de octubre de 2023 7:57

Para crear carpetas en Windows:

Creas un archivo .bat y lo rellenas con lo que quieras que ejecute:

Ejemplo crear carpetas:

```
md M01-FdT_Big_Data M02-blabla\Bibliografía
```

Ejecutar un script de python:

```
@echo off
```

```
@echo off
```

```
"C:\Users\marti\AppData\Local\Programs\Python\Python311\python.exe" "C:\Users\marti\Downloads\Programa\File-Downloads-Automator-main\fileAutomator.py"
```

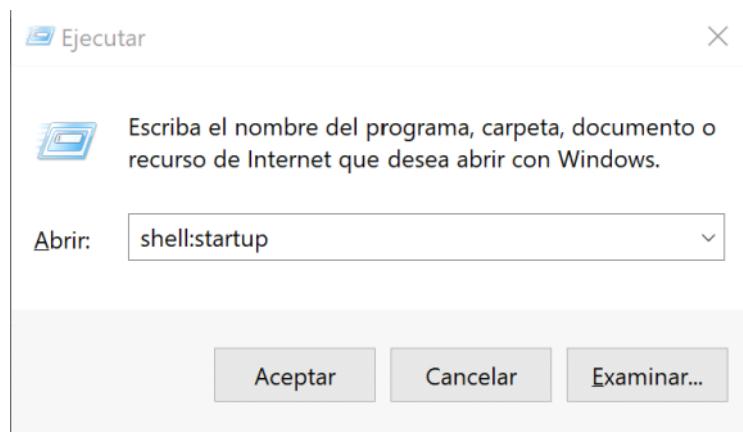


File-Downlo  
ads-Auto...

Para ejecutar al inicio:

Windows + R

shell:startup



## FILEAUTOMATOR PY

miércoles, 25 de octubre de 2023 8:34

Guardar archivos automáticamente según entran en una carpeta, basado en su extensión:

```
from os import scandir, rename
from os.path import splitext, exists, join
from shutil import move
from time import sleep
import logging
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
# ! FILL IN BELOW
# ? folder to track e.g. Windows: "C:\\\\Users\\\\UserName\\\\Downloads"
source_dir = "C:\\\\Users\\\\marti\\\\Downloads"
dest_dir_config = "C:\\\\Users\\\\marti\\\\Downloads\\\\Configuration_Files"
dest_dir_sfx = "C:\\\\Users\\\\marti\\\\Downloads\\\\Audio_Files\\\\SFX_Files"
dest_dir_music = "C:\\\\Users\\\\marti\\\\Downloads\\\\Audio_Files"
dest_dir_video = "C:\\\\Users\\\\marti\\\\Downloads\\\\Video_Files"
dest_dir_image = "C:\\\\Users\\\\marti\\\\Downloads\\\\Image_Files"
dest_dir_documents = "C:\\\\Users\\\\marti\\\\Downloads\\\\Document_Files"
# ? supported image types
image_extensions = [".jpg", ".jpeg", ".jpe", ".jfif", ".jfi", ".png", ".gif", ".webp", ".tiff", ".tif", ".psd", ".raw", ".arw", ".cr2", ".nrw",
                     ".k25", ".bmp", ".dib", ".heif", ".heic", ".ind", ".indd", ".indt", ".jp2", ".j2k", ".jpf", ".jpf", ".jpx", ".jpm", ".mj2",
                     ".svg", ".svgz", ".ai", ".eps", ".ico"]
# ? supported Video types
video_extensions = [".webm", ".mpg", ".mp2", ".mpeg", ".mpe", ".mpv", ".ogg",
                     ".mp4", ".mp4v", ".m4v", ".avi", ".wmv", ".mov", ".qt", ".flv", ".swf", ".avchd"]
# ? supported Audio types
audio_extensions = [".m4a", ".flac", ".mp3", ".wav", ".wma", ".aac"]
# ? supported Document types
document_extensions = [".doc", ".docx", ".odt",
                        ".pdf", ".xls", ".xlsx", ".ppt", ".pptx"]
# ? supported Configuration types
config_extensions = [".xml", ".conf", ".LxAddon"]

def make_unique(dest, name):
    filename, extension = splitext(name)
    counter = 1
    # * IF FILE EXISTS, ADDS NUMBER TO THE END OF THE FILENAME
    while exists(f"{dest}/{name}"):
        name = f"{filename}{str(counter)}{extension}"
        counter += 1
    return name

def move_file(dest, entry, name):
    if exists(f"{dest}/{name}"):
        unique_name = make_unique(dest, name)
        oldName = join(dest, name)
        newName = join(dest, unique_name)
        rename(oldName, newName)
    move(entry, dest)

class MoverHandler(FileSystemEventHandler):
    # ? THIS FUNCTION WILL RUN WHENEVER THERE IS A CHANGE IN "source_dir"
    # ? .upper is for not missing out on files with uppercase extensions
    def on_modified(self, event):
        with scandir(source_dir) as entries:
            for entry in entries:
                name = entry.name
                self.check_audio_files(entry, name)
                self.check_video_files(entry, name)
                self.check_image_files(entry, name)
                self.check_document_files(entry, name)
                self.check_config_files(entry, name)
    def check_audio_files(self, entry, name): # * Checks all Audio Files
        for audio_extension in audio_extensions:
            if name.endswith(audio_extension) or name.endswith(audio_extension.upper()):
                if entry.stat().st_size < 10_000_000 or "SFX" in name: # ? 10Megabytes
                    dest = dest_dir_sfx
                else:
                    dest = dest_dir_music
                move_file(dest, entry, name)
                logging.info(f"Moved audio file: {name}")
    def check_video_files(self, entry, name): # * Checks all Video Files
        for video_extension in video_extensions:
            if name.endswith(video_extension) or name.endswith(video_extension.upper()):
                move_file(dest_dir_video, entry, name)
                logging.info(f"Moved video file: {name}")
    def check_image_files(self, entry, name): # * Checks all Image Files
        for image_extension in image_extensions:
            if name.endswith(image_extension) or name.endswith(image_extension.upper()):
                move_file(dest_dir_image, entry, name)
                logging.info(f"Moved image file: {name}")
    def check_document_files(self, entry, name): # * Checks all Document Files
        for documents_extension in document_extensions:
            if name.endswith(documents_extension) or name.endswith(documents_extension.upper()):
                move_file(dest_dir_documents, entry, name)
                logging.info(f"Moved document file: {name}")
    def check_config_files(self, entry, name): # * Checks all Document Files
        for configs_extension in config_extensions:
```

```
if name.endswith(configs_extension) or name.endswith(configs_extension.upper()):  
    move_file(dest_dir_config, entry, name)  
    logging.info(f"Moved configuration file: {name}")  
# ! NO NEED TO CHANGE BELOW CODE  
if __name__ == "__main__":  
    logging.basicConfig(level=logging.INFO,  
                        format='%(asctime)s - %(message)s',  
                        datefmt='%Y-%m-%d %H:%M:%S')  
path = source_dir  
event_handler = MoverHandler()  
observer = Observer()  
observer.schedule(event_handler, path, recursive=True)  
observer.start()  
try:  
    while True:  
        sleep(10)  
except KeyboardInterrupt:  
    observer.stop()  
observer.join()
```

## Descarga Ficheros de WEB

martes, 31 de octubre de 2023 9:17

```
#####
#Descarga de todos los links a los archivos que hay en una web
#según el número de miniserver sin necesidad de conocer la dirección IP gracias al
servicio ddns
#librería para descargar los datos
import requests
from bs4 import BeautifulSoup
#librería para poder crear
import os
import time
miniserver='504F94AXXX'
puerto= 8080

IP="direccion.ddns.net:& puerto& "/stats/"
print(IP)
URL='http://user:password@' + IP
print(URL)
page = requests.get(URL + IP)
#bs = BeautifulSoup(page.content, features='lxml')
ruta='C:/ruta del archivo padre' + '/' + miniserver + '/'
#Comprobar si existe el directorio
if os.path.isdir(ruta):
    print('Existe el directorio')
else:
    os.mkdir(ruta)

#os.mkdir(ruta)
os.chdir(ruta)
print(page.content)
bs = BeautifulSoup(page.content, "html.parser")
for link in bs.findAll('a'):
    urldownload = URL + str(link.get('href'))
    nombre = str(link.text) + '.xml'
    r = requests.get(urldownload, allow_redirects=True)
    time.sleep(1)
    open(nombre.replace(":", "-"), 'wb').write(r.content)
```

# Data Analysis Future

lunes, 6 de noviembre de 2023 9:02

[Simplifying Data Analysis - Live from PyData NYC 2023](#)



Series temporales gratis:  
Influxdb (base de datos)

Para graficarlo: grafana

# GOALS

miércoles, 30 de octubre de 2024 2:56

## 1. Data Management: Basic Concepts and Fundamentals

This section will focus on understanding the lifecycle of data, from collection to storage and retrieval.

- **Data Lifecycle:** Understanding data creation, data curation, and data retirement.
- **Data Governance:** Policies, procedures, and standards around data management.
- **Data Quality:** Ensuring accuracy, consistency, and completeness of data.
- **Database Management Systems:** Overview of traditional and NoSQL databases.

# Materials

lunes, 28 de octubre de 2024 18:23

[Is data management the secret to generative AI?](#)



# SESSION I

miércoles, 30 de octubre de 2024 2:34

## Session 1: Introduction to Data Management

**Objective:** Introduce students to the foundational concepts of data management, emphasizing the role of data in organizations and the basics of data types, structures, and storage. This session aims to provide a comprehensive understanding of how data is organized, accessed, and its significance in decision-making processes.

### *Session Outline (2 Hours)*

| Section  | Time   | Content  | Activities   |
|--|--------|--|--|
| <b>1. Introduction to Data</b>                         | 15 min | - Overview of data as an asset in organizations.<br>- Importance of data-driven decisions.                                   | Group discussion: "Why is data valuable?"  |
| <b>2. Key Concepts in Data Management</b>              | 20 min | - Data lifecycle (creation, storage, usage, deletion).<br>- Overview of data quality, data privacy, and compliance.          | Interactive Q&A on data lifecycle steps  |
| <b>3. Types of Data</b>                                | 25 min | - Structured, unstructured, and semi-structured data.<br>- Examples of each type (tables, images, logs).                     | Case examples: Identify data types in transportation data                              |
| <b>4. Data Storage Systems</b>                         | 20 min | - Overview of databases, data warehouses, and data lakes.<br>- Differences and applications in business contexts.            | Visual comparison exercise: Database vs. Data Lake                                     |
| <b>5. Introduction to Data Management Technologies</b> | 20 min | - Common tools and systems (SQL databases, NoSQL, data lakes).<br>- Brief introduction to SQL vs. NoSQL databases.           | Group brainstorm: Discuss potential uses of each technology                            |
| <b>6. Case Study: Transportation Data Management</b>   | 15 min | - Using transportation data to understand data organization.<br>- Discuss data sources, types, and usage in decision-making. | Group activity: Outline a simple data structure for geolocation and working hours data |
| <b>7. Wrap-Up and Q&amp;A</b>                          | 5 min  | - Recap key points.<br>- Address any questions.  | -  |

# Data Collection

miércoles, 30 de octubre de 2024 13:08

## Sources

- Central Banks
- Weather
- Google search
- IOT : sensors (vibration, sound)
- CRM
- Web Scraping
- Call Centers
- Customer Databases
- Sales Records
- Social Media

# ACTIVITY

miércoles, 30 de octubre de 2024 2:41

Access control in a business that will be used to control working hours, among other things.  
**Set up a Data Management System to store, secure, access and visualize this data.**

[https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis/getting-started-pvgis/api-non-interactive-service\\_en?prefLang=mt](https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis/getting-started-pvgis/api-non-interactive-service_en?prefLang=mt)

# Python

jueves, 7 de noviembre de 2024 8:23

## 5 Useful f String Tricks



Create Virtual Environment:

python -m venv myenv

What libraries I have installed:

pip freeze

Version of python:

python --version

pip --version

Create a virtual environment

Activation

source Python Examples\myenv\Scripts\activate

<https://riteshshergill.medium.com/creating-amazing-visualizations-with-matplotlib-and-seaborn-486fa7d68c1e>

# Sources

jueves, 7 de noviembre de 2024 12:31

Files:

Csv, json, xlm, xls, jpeg....

Databases:

Sql

NoSql

API

Streaming

Web Scraping

# POWERQUERY

lunes, 18 de noviembre de 2024 11:25

CALENDAR TABLE:

```
let
    /*
    ****This Calendar was created and provided by Avi Singh****
    ****This can be freely shared as long as this text comment is retained.****
    http://www.youtube.com/PowerBIPro
    www.LearnPowerBI.com by Avi Singh
    */
    #"LearnPowerBI.com by Avi Singh" = 1,
    StartDate = #date(2018, 1, 1),
    EndDate = Date.EndOfYear(DateTime.Date(DateTime.FixedLocalNow())) /*was "#date(2017, 1, 1)"
Updated on 201802027: hard Coded End of Year caused some formulas to break, switching to
dynamic date*/,
    //Used for 'Offset' Column calculations, you may Hard code CurrentDate for testing e.g.
    #date(2017,9,1)
    CurrentDate = DateTime.Date(DateTime.FixedLocalNow()),
    // Specify the last month in your Fiscal Year, e.g. if June is the last month of your Fiscal Year,
    specify 6
    FiscalYearEndMonth = 6,
    #"==SET PARAMETERS ABOVE==" = 1,
    #"==Build Date Column==" = #"==SET PARAMETERS ABOVE==",
    ListDates = List.Dates(StartDate, Number.From(EndDate - StartDate)+1, #duration(1,0,0,0)),
    #"Converted to Table" = Table.FromList(ListDates, Splitter.SplitByNothing(), null, null,
    ExtraValues.Error),
    #"Renamed Columns as Date" = Table.RenameColumns(#"Converted to Table",{{"Column1",
    "Date"}}),
    // As far as Power BI is concerned, the 'Date' column is all that is needed :-) But we will continue
    and add a few Human-Friendly Columns
    #"Changed Type to Date" = Table.TransformColumnTypes(#"Renamed Columns as Date",{{"Date",
    type date}}),
    #"==Add Calendar Columns==" = #"Changed Type to Date",
    #"Added Calendar MonthNum" = Table.AddColumn(#"==Add Calendar Columns==", "MonthNum",
    each Date.Month([Date]), Int64.Type),
    #"Added Month Name" = Table.AddColumn(#"Added Calendar MonthNum", "Month", each
    Text.Start(Date.MonthName([Date],"es-ES"),3)),
    #"Added Month Name Long" = Table.AddColumn(#"Added Month Name", "MonthLong", each
    Date.MonthName([Date], "es-ES"), type text),
    #"Added Calendar Quarter" = Table.AddColumn(#"Added Month Name Long", "Quarter", each "Q"
    & Text.From(Date.QuarterOfYear([Date]))),
    #"Added Calendar Year" = Table.AddColumn(#"Added Calendar Quarter", "Year", each
```

```

Date.Year([Date]), Int64.Type),
#"==Add Fiscal Calendar Columns==" = #"Added Calendar Year",
#"Added FiscalMonthNum" = Table.AddColumn(#"==Add Fiscal Calendar Columns==",
"FiscalMonthNum", each if [MonthNum] > FiscalYearEndMonth
then [MonthNum] - FiscalYearEndMonth
else [MonthNum] + (12 - FiscalYearEndMonth), type number),
#"Added FiscalMonth Name" = Table.AddColumn(#"Added FiscalMonthNum", "FiscalMonth", each
[Month]),
#"Added FiscalMonth Name Long" = Table.AddColumn(#"Added FiscalMonth Name",
"FiscalMonthLong", each [MonthLong]),
#"Added FiscalQuarter" = Table.AddColumn(#"Added FiscalMonth Name Long", "FiscalQuarter",
each "FQ" & Text.From(Number.RoundUp([FiscalMonthNum] / 3,0))),
#"Added FiscalYear" = Table.AddColumn(#"Added FiscalQuarter", "FiscalYear", each "FY" &
Text.End(
Text.From(
if [MonthNum] > FiscalYearEndMonth
then [Year] + 1
else [Year]
)
,2
)),
#"==Add Calendar Date Offset Columns==" = #"Added FiscalYear",
// Can be used to for example to show the past 3 months(CurMonthOffset = 0, -1, -2)
#"Added CurMonthOffset" = Table.AddColumn(#"==Add Calendar Date Offset Columns==",
"CurMonthOffset", each ( Date.Year([Date]) - Date.Year(FromDate) ) * 12
+ Date.Month([Date]) - Date.Month(FromDate), Int64.Type),
// Can be used to for example to show the past 3 quarters (CurQuarterOffset = 0, -1, -2)
#"Added CurQuarterOffset" = Table.AddColumn(#"Added CurMonthOffset", "CurQuarterOffset",
each /*Year Difference*/
( Date.Year([Date]) - Date.Year(FromDate) )*4
/*Quarter Difference*/
+ Number.RoundUp(Date.Month([Date]) / 3)
- Number.RoundUp(Date.Month(FromDate) / 3),
Int64.Type),
// Can be used to for example to show the past 3 years (CurYearOffset = 0, -1, -2)
#"Added CurYearOffset" = Table.AddColumn(#"Added CurQuarterOffset", "CurYearOffset", each
Date.Year([Date]) - Date.Year(FromDate), Int64.Type),
// Can be used to for example filter out all future dates
#"Added FutureDate Flag" = Table.AddColumn(#"Added CurYearOffset", "FutureDate", each if
[Date] > CurrentDate then "Future" else "Past" ),
// FiscalYearOffset is the only Offset that is different.
// FiscalQuarterOffset = is same as CurQuarterOffset

```

```

// FiscalMonthOffset = is same as CurMonthOffset
#"==Add FiscalYearOffset==" = #"Added FutureDate Flag",
#"Filtered Rows to CurrentDate" = Table.SelectRows(#"==Add FiscalYearOffset==", each ([Date] = CurrentDate)),
CurrentFiscalYear = #"Filtered Rows to CurrentDate">{0}[FiscalYear],
#"Continue...Orig Table" = #"==Add FiscalYearOffset==",
#"Added CurFiscalYearOffset" = Table.AddColumn(#"Continue...Orig Table", "CurFiscalYearOffset", each Number.From(Text.Range([FiscalYear],2,2)) -
Number.From(Text.Range(CurrentFiscalYear,2,2)))
/*Extract the numerical portion, e.g. FY18 = 18*/,
#"==Add General Columns==" = #"Added CurFiscalYearOffset",
// Used as 'Sort by Column' for MonthYear columns
#"Added MonthYearNum" = Table.AddColumn(#"==Add General Columns==", "MesAñoNum", each [Year]*100 + [MonthNum] /*e.g. Sep-2016 would become 201609*/, Int64.Type),
#"Added MonthYear" = Table.AddColumn(#"Added MonthYearNum", "MonthYear", each [Month] & "-" & Text.End(Text.From([Year]),2)),
#"Added MonthYearLong" = Table.AddColumn(#"Added MonthYear", "MonthYearLong", each [Month] & "-" & Text.From([Year])),
#"Added WeekdayNum" = Table.AddColumn(#"Added MonthYearLong", "WeekdayNum", each Date.DayOfWeek([Date],Day.Monday)+1),
#"Added Weekday Name" = Table.AddColumn(#"Added WeekdayNum", "Weekday", each Text.Start(Date.DayOfWeekName([Date],"es-ES"),3)),
#"Added WeekdayWeekend" = Table.AddColumn(#"Added Weekday Name", "WeekdayWeekend", each if [WeekdayNum] = 6 or [WeekdayNum] = 7 then "Weekend"
else "Weekday"),
#"==Improve Ultimate Table" = #"Added WeekdayWeekend",
#"----Add WeekSequenceNum----" = #"==Improve Ultimate Table",
#"Filtered Rows Sundays Only (Start of Week)" = Table.SelectRows(#"----Add WeekSequenceNum----", each [WeekdayNum] = 7),
#"Added Index WeekSequenceNum" = Table.AddIndexColumn(#"Filtered Rows Sundays Only (Start of Week)", "WeekSequenceNum", 2, 1),
#"Merged Queries Ultimate Table to WeekSequenceNum" = Table.NestedJoin(#"==Improve Ultimate Table", {"Date"}, {"Added Index WeekSequenceNum", {"Date"}, "Added Index WeekNum"}, JoinKind.LeftOuter),
#"Expanded Added Index WeekNum" = Table.ExpandTableColumn(#"Merged Queries Ultimate Table to WeekSequenceNum", "Added Index WeekNum", {"WeekSequenceNum"}, {"WeekSequenceNum"}),
// somehow it ends up being unsorted after Expand Column, should not matter for the end table, but makes it harder to debug and check everything is correct. Thus sorting it.
#"ReSorted Rows by Date" = Table.Sort(#"Expanded Added Index WeekNum", {"Date", Order.Ascending}),
#"Filled Down WeekSequenceNum" = Table.FillDown(#"ReSorted Rows by Date", {"WeekSequenceNum"}),
#"Replaced Value WeekSequenceNum null with 1" = Table.ReplaceValue(#"Filled Down WeekSequenceNum", null, 1, Replacer.ReplaceValue, {"WeekSequenceNum"}),

```

```
#"----WeekSequenceNum Complete----" = #"Replaced Value WeekSequenceNum null with 1",
Current_WeekSequenceNum =#"----WeekSequenceNum Complete----" {[Date =
CurrentDate]}?[WeekSequenceNum],
#"Added Custom CurWeekOffset" = Table.AddColumn(#"----WeekSequenceNum Complete----",
"CurWeekOffset", each [WeekSequenceNum] - Current_WeekSequenceNum, Int64.Type),
#"Semana del año insertada" = Table.AddColumn(#"Added Custom CurWeekOffset", "Semana del
año", each Date.WeekOfYear([Date]), Int64.Type),
#"Filas filtradas" = Table.SelectRows(#"Semana del año insertada", each true)

in

#"Filas filtradas"
```

# CALENDAR

lunes, 18 de noviembre de 2024 11:40

```
let
    StartDate = #date(2022, 1, 1),
    EndDate = #date(2023, 12, 31),
    DateList = List.Dates(StartDate, Duration.Days(EndDate - StartDate) + 1, #duration(1, 0, 0)),
    CalendarTable = Table.FromList(DateList, Splitter.SplitByNothing(), {"Date"}),
    #"Personalizado agregado" =
        Table.TransformColumnTypes(Table.AddColumn(CalendarTable, "Year", each Date.Year([Date])), {"Year", Int64.Type}),
    #"Personalizado agregado 1" =
        Table.TransformColumnTypes(Table.AddColumn(#"Personalizado agregado", "Month", each Date.MonthName([Date])), {"Month", type text}),
    #"Personalizado agregado 2" =
        Table.TransformColumnTypes(Table.AddColumn(#"Personalizado agregado 1", "Month Number", each Date.Month([Date])), {"Month Number", Int64.Type}),
    #"Columnas con nombre cambiado" = Table.RenameColumns(#"Personalizado agregado 2", {"Month", "Month Name"}),
    #"Personalizado agregado 3" =
        Table.TransformColumnTypes(Table.AddColumn(#"Columnas con nombre cambiado", "Day", each Date.Day([Date])), {"Day", Int64.Type}),
    #"Tipo de columna cambiado" = Table.TransformColumnTypes(#"Personalizado agregado 3", {"Date", type date})
in
    #"Tipo de columna cambiado"
```

Calendar

Date  
Number days since begining  
Day  
Month  
Number of week (ISO)  
Fiscal year  
Calendar year

# SQL

sábado, 9 de noviembre de 2024 5:57

Chartdb.io

We are going to review what is an SQL database management system, how to create a database, operate with it in the most common ways.

We have selected as SQL database, PostgreSQL

1st: create a user

2nd: create a database

3rd: introduce the basic commands: create, read, update and delete data. Commonly referred as **CRUD**

4th: import data

5th: create relationships between tables

6th: perform queries to retrieve data

# Example

miércoles, 20 de noviembre de 2024 6:03

I have a list of vehicles I would like to check if that vehicle is allowed to enter the facilities or not. APNR camera is collecting numberplates automatically.

When a numberplate is collected, the barrier opens if the numberplate is allowed. This information is saved and stored in the camera, and it can be exported to an .xls file.

If it is not in the list, or it is blocked to enter, the car should stop at the sign and give their data to be registered in the system.

## Tasks:

PostgreSQL database (install)

Create tables:

- List of Authorized Vehicles (import data from .xls file)
- Records of Numberplate registrations each time a numberplate is recorded

Create a connection between tables, so that we can

Prompt:

Understood! I'll simplify the column names while keeping them meaningful and aligned with the original data structure. Here's the updated script:

## Simplified Column Names and Script

```
python
Copiar código
import pandas as pd
import psycopg2
import os
# Path to the .xls file and the output .csv
xls_file_path = r"C:\Users\marti\OneDrive - Ensaco\09.Desarrollos\ZORONGO OFICINA\MATRÍCULAS
\plateNolist_0.0.0.0_20241120054625.xls"
csv_file_path = os.path.splitext(xls_file_path)[0] + ".csv" # Generate the CSV path with the same
base name
# Read the .xls file
data = pd.read_excel(xls_file_path)
# Save as .csv in the same folder
data.to_csv(csv_file_path, index=False, sep='\t')
print(f"CSV file generated at: {csv_file_path}")
# Connect to the PostgreSQL database
conn = psycopg2.connect(
    dbname="zorongo",
    user="your_user", # Replace with your PostgreSQL username
```

```

password="your_password", # Replace with your PostgreSQL password
host="localhost",
port="5432"
)
cursor = conn.cursor()
# Create the table with simplified column names if it doesn't exist
create_table_query = """
CREATE TABLE IF NOT EXISTS vehiculos_autORIZADOS_visitantes (
    id SERIAL PRIMARY KEY,
    plate_no VARCHAR(20) NOT NULL,
    group_type INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    card_id VARCHAR(50)
);
"""

cursor.execute(create_table_query)
conn.commit()
print("Table checked or created successfully.")
# Insert data from the CSV
with open(csv_file_path, 'r', encoding='utf-8') as csv_file:
    cursor.copy_expert(f"""
        COPY vehiculos_autORIZADOS_visitantes (
            plate_no,
            group_type,
            start_date,
            end_date,
            card_id
        )
        FROM STDIN
        DELIMITER '\t'
        CSV HEADER;
    """, csv_file)
conn.commit()
cursor.close()
conn.close()
print("Data successfully imported into PostgreSQL.")

```

## Changes Made

1. Simplified Column Names:
  - Original column names were replaced as follows:
    - "Plate No." → plate\_no
    - "Group(0 BlockList, 1 AllowList)" → group\_type
    - "Effective Start Date (Format: YYYY-MM-DD, e.g., 2017-12-07)" → start\_date
    - "Effective End Date (Format: YYYY-MM-DD, e.g., 2017-12-07)" → end\_date
    - "Card ID" → card\_id
2. Table Creation Logic:
  - The script creates a table with these simplified column names and ensures compatibility with the provided data types.
3. CSV Processing:
  - The .xls file is converted into a .csv file using the same name, saved in the same folder as the original file.
4. Database Import:
  - The COPY command uses the simplified column names to load the data efficiently.

## Example Table Schema in PostgreSQL

After running the script, the table structure will look like this:

```
sql
Copiar código
CREATE TABLE vehiculos_autorizados_visitantes (
    id SERIAL PRIMARY KEY,
    plate_no VARCHAR(20) NOT NULL,
    group_type INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    card_id VARCHAR(50)
);
```

## What You Need to Do:

1. Replace your\_user and your\_password with your PostgreSQL credentials.
2. Run the script in Python.
3. Verify the data import with:

```
sql
Copiar código
SELECT * FROM vehiculos_autorizados_visitantes;
```

Let me know if further refinements are needed!

# POSTGRES INSTALLATION GUIDE

miércoles, 20 de noviembre de 2024 6:50

Para añadir psql al PATH en Windows y poder usarlo desde la línea de comandos, sigue estos pasos:

## 1. Encuentra la ubicación de psql.exe

Busca en la carpeta de instalación de PostgreSQL. Generalmente, se encuentra en una ruta como:  
makefile

Copiar código

C:\Program Files\PostgreSQL\<versión>\bin

Dentro de esta carpeta, encontrarás psql.exe.

## 2. Añade la ruta al PATH

Opción 1: Desde la interfaz gráfica

Haz clic derecho en el botón de Inicio y selecciona Sistema o Configuración del sistema.

Ve a Configuración avanzada del sistema (generalmente en el panel lateral derecho).

En la ventana que se abre, haz clic en el botón Variables de entorno.

En la sección Variables del sistema, selecciona la variable llamada Path y haz clic en Editar.

Haz clic en Nuevo e introduce la ruta donde se encuentra psql.exe, por ejemplo:

makefile

Copiar código

C:\Program Files\PostgreSQL\<versión>\bin

Haz clic en Aceptar en todas las ventanas para guardar los cambios.

Opción 2: Desde PowerShell

Abre PowerShell como administrador.

Ejecuta el siguiente comando para añadir la ruta:

powershell

Copiar código

```
[Environment]::SetEnvironmentVariable("Path", $([Environment]::GetEnvironmentVariable("Path", "Machine") + ";C:\Program Files\PostgreSQL\<versión>\bin"), "Machine")
```

Reemplaza <versión> con el número correspondiente de tu instalación.

## 3. Verifica el cambio

Abre una nueva ventana del Símbolo del sistema (CMD) o PowerShell.

Escribe:

cmd

Copiar código

psql --version

Si se muestra la versión de psql, el cambio se realizó correctamente.

¡Ahora puedes usar psql desde cualquier terminal de Windows!

# TUTORIAL BASICS

miércoles, 20 de noviembre de 2024 6:51

## Tutorial: Introducción a PostgreSQL para Universitarios

### Requisitos previos

#### 1. Instalar PostgreSQL:

##### o Windows:

1. Descarga el instalador desde [PostgreSQL.org](https://www.postgresql.org).
2. Sigue el asistente de instalación y asegúrate de:
  - Seleccionar pgAdmin como herramienta visual.
  - Configurar una contraseña para el usuario postgres.

##### o Mac:

1. Instala PostgreSQL usando [Postgres.app](https://www.postgres.app) o brew:

```
bash
Copiar código
brew install postgresql
```

2. Inicia el servicio:

```
bash
Copiar código
brew services start postgresql
```

#### 2. Instalar un cliente SQL (opcional si no usas pgAdmin):

- [DBeaver](#) es multiplataforma.

#### 3. Configurar PATH (para usar PostgreSQL desde la terminal):

- Windows**: Agrega el directorio bin de PostgreSQL a las variables de entorno.
- Mac**: PostgreSQL suele añadirlo automáticamente al PATH.

## Paso 1: Crear un Usuario

Desde la terminal (cmd en Windows o Terminal en Mac):

#### 1. Acceder a PostgreSQL:

```
bash
Copiar código
psql -U postgres
```

#### 2. Crear un usuario:

```
sql
Copiar código
CREATE USER myuser WITH PASSWORD 'mypassword';
```

#### 3. Permitir creación de bases de datos:

```
sql
Copiar código
ALTER USER myuser WITH CREATEDB;
```

## Paso 2: Crear una Base de Datos

#### 1. Conectar como el usuario creado:

```
bash
Copiar código
```

```
psql -U myuser
```

## 2. Crear la base de datos:

```
sql
```

Copiar código

```
CREATE DATABASE university_db;
```

## 3. Conectar a la base de datos:

```
bash
```

Copiar código

```
\c university_db
```

# Paso 3: Operaciones CRUD Básicas

## Crear una Tabla

```
sql
```

Copiar código

```
CREATE TABLE students (
```

```
    id SERIAL PRIMARY KEY,
```

```
    name VARCHAR(100),
```

```
    age INT,
```

```
    major VARCHAR(50)
```

```
);
```

## Insertar Datos

```
sql
```

Copiar código

```
INSERT INTO students (name, age, major) VALUES
```

```
('Alice', 20, 'Computer Science'),
```

```
('Bob', 22, 'Mathematics');
```

## Leer Datos

```
sql
```

Copiar código

```
SELECT * FROM students;
```

## Actualizar Datos

```
sql
```

Copiar código

```
UPDATE students SET age = 21 WHERE name = 'Alice';
```

## Eliminar Datos

```
sql
```

Copiar código

```
DELETE FROM students WHERE name = 'Bob';
```

# Paso 4: Importar Datos

## 1. Crear un archivo CSV (students.csv):

```
csv
```

Copiar código

```
id,name,age,major
```

```
1,John,23,Physics
```

```
2,Lisa,19,Biology
```

## 2. Cargar el archivo en la tabla:

- **Windows:** Verifica la ruta completa del archivo.

- **Mac:** Usa ~/ para rutas locales.

```
sql
Copiar código
\COPY students FROM '/ruta/al/archivo/students.csv' DELIMITER ',' CSV HEADER;
```

## Paso 5: Relaciones entre Tablas

### 1. Crear una Tabla Relacionada:

```
sql
Copiar código
CREATE TABLE courses (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100)
);
```

### 2. Agregar una Clave Foránea:

```
sql
Copiar código
ALTER TABLE students ADD COLUMN course_id INT REFERENCES courses(id);
```

### 3. Insertar y Vincular Datos:

```
sql
Copiar código
INSERT INTO courses (name) VALUES ('Database Systems');
UPDATE students SET course_id = 1 WHERE name = 'Alice';
```

## Paso 6: Consultas Relacionales

### 1. Consulta con JOIN:

```
sql
Copiar código
SELECT students.name, students.age, courses.name AS course
FROM students
JOIN courses ON students.course_id = courses.id;
```

### 2. Agrupaciones y Conteos:

```
sql
Copiar código
SELECT courses.name, COUNT(students.id) AS student_count
FROM students
JOIN courses ON students.course_id = courses.id
GROUP BY courses.name;
```

## Notas Finales

- **Práctica sugerida:** Pide a los estudiantes que creen su propia base de datos para gestionar información de un proyecto (por ejemplo, una biblioteca o sistema de gestión de notas).
- **Consejo:** Introducir herramientas visuales como pgAdmin o DBeaver para facilitar la navegación a quienes no se sientan cómodos con la terminal.

Este tutorial ofrece una base sólida para que los estudiantes comprendan PostgreSQL y su funcionalidad. ¿Necesitas añadir ejercicios o algún enfoque especial?

# Md and VSCode

miércoles, 20 de noviembre de 2024 6:57



## PostgreSQL \_Tutorial

Teaching a PostgreSQL class using Visual Studio Code (VS Code) can be a great way to combine the Markdown file as a guide and use extensions to run PostgreSQL commands directly. Here's how you can set it up:

### 1. Install the Necessary Extensions

#### 1. Markdown Preview:

- Install the "**Markdown Preview Enhanced**" extension to view the Markdown file while teaching.
- Open your Markdown file and use the shortcut Ctrl+Shift+V (Windows/Linux) or Cmd+Shift+V (Mac) to preview it.

#### 2. SQL Tools:

- Install the "**SQLTools**" extension.
- Install the "**SQLTools PostgreSQL/Redshift Driver**" plugin to connect to PostgreSQL.

### 2. Set Up PostgreSQL in VS Code

#### 1. Configure SQLTools:

- Open the command palette (Ctrl+Shift+P or Cmd+Shift+P) and search for SQLTools: Add New Connection.
- Select **PostgreSQL** and provide the following details:
  - **Server Address:** localhost (or the server you're using)
  - **Port:** 5432 (default for PostgreSQL)
  - **User:** Your PostgreSQL user (e.g., postgres or myuser)
  - **Password:** The password for the user.
  - **Database:** Leave it blank to list all databases or specify the one you'll work on.
- Test the connection and save.

#### 2. Run SQL Queries:

- Open a new .sql file in VS Code.
- Use the shortcut Ctrl+Alt+E (Windows/Linux) or Cmd+Option+E (Mac) to execute selected queries.

### 3. Split Screen for Markdown and SQL

- Open the Markdown file on the left and enable preview mode for students to follow the guide.
- Open an .sql file on the right to run commands.

### 4. Enable PostgreSQL Logs

For teaching purposes, you can show logs of what happens when queries are run. Use the psql terminal or check PostgreSQL logs.

### 5. Enhance Interactivity

- Use **Live Share** (a VS Code extension) to allow students to follow your edits or even collaborate in real time.
- Share the screen for the Markdown preview or use a projector.

## 6. Optional: PostgreSQL CLI Integration

If you want to demonstrate PostgreSQL commands directly in the terminal:

1. Open the integrated terminal in VS Code (`Ctrl+ or Cmd+).
2. Run PostgreSQL commands using psql:

```
bash  
Copiar código  
psql -U postgres
```

This setup lets you seamlessly follow the tutorial, run live SQL queries, and keep your class engaged with a clear structure. Let me know if you'd like additional help setting it up!

# EXAMPLE VEHICLE LOG ENTRY

miércoles, 20 de noviembre de 2024 8:05

Here's how we can create and explain a project for a License Plate Entry Log system in a university class, using a relational database. The project will have two tables as described: one for visitor entries and another for authorized/unauthorized vehicles.

## Project Overview

Goal:

Build a database to track visitor entries and manage a registry of authorized and unauthorized vehicles.

Learn how to create tables, relationships, and constraints.

Demonstrate the importance of foreign key relationships and how to handle missing references.

## Step-by-Step Explanation

### 1. Database Schema

We'll design two tables:

VisitorsLog: Logs every visitor's entry with details such as timestamp, license plate, reason for the visit, and destination.

VehiclesRegistry: Maintains a record of authorized and unauthorized vehicles.

### 2. Tables and Relationships

The VisitorsLog table will include:

LogID: A unique identifier for each entry (primary key).

Timestamp: Date and time of entry, set automatically.

LicensePlate: The license plate of the vehicle (links to the VehiclesRegistry table).

Reason: A description of the visitor's purpose (can be NULL).

Destination: The destination within the urbanization (can be NULL).

The VehiclesRegistry table will include:

LicensePlate: The unique license plate of the vehicle (primary key).

Status: Indicates whether the vehicle is authorized or unauthorized.

### 3. SQL Script to Create the Tables

Here's the SQL code to create the schema:

sql

Copiar código

-- Table to log visitor entries

CREATE TABLE VisitorsLog (

LogID SERIAL PRIMARY KEY,

Timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL,

LicensePlate VARCHAR(20) NOT NULL,

Reason TEXT NULL,

Destination TEXT NULL,

FOREIGN KEY (LicensePlate) REFERENCES VehiclesRegistry(LicensePlate)

ON DELETE SET NULL

);

-- Table for the vehicles registry

CREATE TABLE VehiclesRegistry (

LicensePlate VARCHAR(20) PRIMARY KEY,

Status VARCHAR(20) NOT NULL CHECK (Status IN ('Authorized', 'Unauthorized'))

);

## Explanation of the Code

### VisitorsLog Table:

LogID: A unique identifier for each visitor's entry. Automatically generated.

Timestamp: Captures the exact date and time of entry. Automatically filled using CURRENT\_TIMESTAMP.

LicensePlate: Refers to the license plate of the vehicle. This column is linked to the LicensePlate column in the VehiclesRegistry table.

Reason and Destination: These fields allow NULL values as they are optional.

### VehiclesRegistry Table:

LicensePlate: A unique identifier for each vehicle. It's the primary key.

Status: Specifies whether the vehicle is Authorized or Unauthorized. The CHECK constraint ensures only these two values can be stored.

### Foreign Key Constraint:

In the VisitorsLog table, the LicensePlate column is a foreign key referencing the LicensePlate column in VehiclesRegistry.

If a referenced vehicle is deleted, the foreign key constraint sets the LicensePlate to NULL in the VisitorsLog table (ON DELETE SET NULL).

### Handling Vehicles Not in the Registry

For entries where the LicensePlate in the VisitorsLog table doesn't exist in the VehiclesRegistry, the foreign key will enforce referential integrity by preventing invalid entries unless explicitly allowed (e.g., nullable reference).

To handle this scenario programmatically, we can:

Insert the visitor's entry regardless of their registry status.

Use a query to identify visitors whose LicensePlate is missing from the VehiclesRegistry.

Example query:

```
sql
Copiar código
SELECT
    v.LogID,
    v.LicensePlate,
    v.Timestamp,
    v.Reason,
    v.Destination,
    CASE
        WHEN vr.LicensePlate IS NULL THEN 'Not Registered'
        ELSE vr.Status
    END AS VehicleStatus
FROM
    VisitorsLog v
LEFT JOIN
    VehiclesRegistry vr ON v.LicensePlate = vr.LicensePlate;
```

Example Workflow for Students

Insert Sample Data:

```
sql
Copiar código
-- Add vehicles to the registry
INSERT INTO VehiclesRegistry (LicensePlate, Status)
VALUES
    ('ABC123', 'Authorized'),
    ('XYZ789', 'Unauthorized');
```

```
-- Log visitor entries
INSERT INTO VisitorsLog (LicensePlate, Reason, Destination)
VALUES
    ('ABC123', 'Delivery', 'House 15'),
    ('XYZ789', 'Visiting Friend', 'House 20'),
    ('UNREG456', 'Inspection', NULL);
```

Run the Query to Analyze Data: Use the provided query to view the visitor logs and identify vehicles not registered.

Expected Output:

yaml

Copiar código

| LogID | LicensePlate | Timestamp           | Reason          | Destination | VehicleStatus  |
|-------|--------------|---------------------|-----------------|-------------|----------------|
| 1     | ABC123       | 2024-11-20 10:00:00 | Delivery        | House 15    | Authorized     |
| 2     | XYZ789       | 2024-11-20 11:00:00 | Visiting Friend | House 20    | Unauthorized   |
| 3     | UNREG456     | 2024-11-20 12:00:00 | Inspection      | NULL        | Not Registered |

Takeaways for Students

Relational Integrity: Foreign keys enforce data consistency.

Handling Missing Data: Use queries to highlight gaps in the dataset.

Real-World Application: Demonstrates a practical use case for databases in managing visitor logs and vehicle access.

Let me know if you'd like additional clarification or more SQL examples for your class! 

# Install psql in raspberry pi

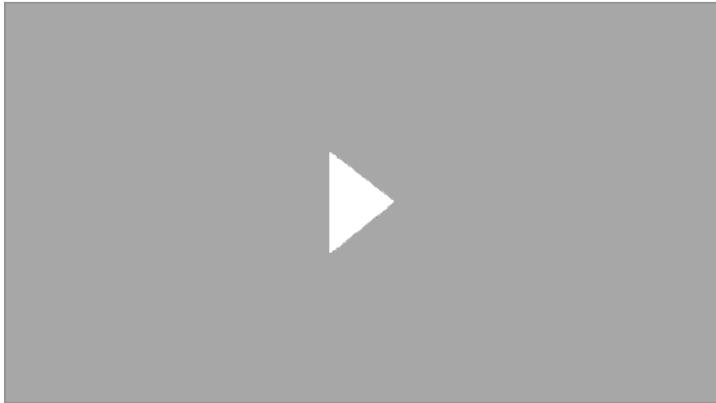
miércoles, 20 de noviembre de 2024 13:32

<https://pimylifeup.com/raspberry-pi-postgresql/>

# NETWORKING

miércoles, 20 de noviembre de 2024 12:46

[what is an IP Address? // You SUCK at Subnetting // EP 1](#)



Ccna (CISCO)

COMPTIA

# HADOOP - read/write examples

lunes, 28 de octubre de 2024 18:23

[HDFS Tutorial For Beginners | HDFS Architecture | HDFS Tutorial | Hadoop Tutorial | Simplilearn](#)



# BIG DATA INSTALATIONS

lunes, 30 de mayo de 2022 7:17

# Install Hadoop

domingo, 29 de mayo de 2022 16:34



## Datos adjuntos

The screenshot shows a Medium article page. At the top, there's a navigation bar with icons for home, notifications, and search. Below it is a sidebar with a PDF icon and other navigation links. The main content area features a dark header with the author's name, João Torres, and the date, Feb 3, 2020, followed by a '6 min read' indicator. A black banner at the top of the content area states: 'To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.' The title of the article is 'How To Set Up a Hadoop 3.2.1 Multi-Node Cluster on Ubuntu 18.04 (2 Nodes)'. Below the title is a large Apache Hadoop logo featuring a yellow elephant and the text 'APACHE hadoop'. The article content starts with a section titled 'To start: What is Hadoop?'. It describes Hadoop as a collection of open-source software utilities for distributed storage and processing of large amounts of data sets across clusters of computers using simple programming models. It also notes its use in MapReduce and its design for computer clusters. The sidebar on the right includes a profile picture of João Torres, his follower count (56), and buttons for 'Follow' and 'Email'. It also lists 'More from Medium' with links to other articles by different authors.

[https://medium.com/@jootorres\\_11979/how-to-set-up-a-hadoop-3-2-1-multi-node-cluster-on-ubuntu-18-04-2-nodes-567ca44a3b12](https://medium.com/@jootorres_11979/how-to-set-up-a-hadoop-3-2-1-multi-node-cluster-on-ubuntu-18-04-2-nodes-567ca44a3b12)

# Install Spark in hadoop multicluster

domingo, 29 de mayo de 2022 16:30

[https://medium.com/@jootorres\\_11979/how-to-install-and-set-up-an-apache-spark-cluster-on-hadoop-18-04-b4d70650ed42](https://medium.com/@jootorres_11979/how-to-install-and-set-up-an-apache-spark-cluster-on-hadoop-18-04-b4d70650ed42)

# VIU-Hadoop\_con\_Docker

domingo, 12 de noviembre de 2023 16:40



VIU-Hadoop  
p\_con\_Do...

# Despliegue de un clúster mediante contenedores Docker

Tomás Fernández Pena

Universidad Internacional de Valencia

Material bajo licencia Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

universidadvalencia

[Componentes de Hadoop](#) [HDFS: Hadoop Distributed File System](#) [Gestión de recursos con YARN](#) [Programación MapReduce](#) [Automatización del despliegue con Docker Compose](#)

## Índice

- 1 Componentes de Hadoop**
- 2 HDFS: Hadoop Distributed File System**  
Despliegue del servicio NameNode
- 3 Gestión de recursos con YARN**  
Despliegue del servicio ResourceManager
- 4 Programación MapReduce**  
Despliegue de los DataNodes/NodeManagers
- 5 Automatización del despliegue con Docker Compose**

## Índice

### 1 Componentes de Hadoop

#### 2 HDFS: Hadoop Distributed File System

Despliegue del servicio NameNode

#### 3 Gestión de recursos con YARN

Despliegue del servicio ResourceManager

#### 4 Programación MapReduce

Despliegue de los DataNodes/NodeManagers

#### 5 Automatización del despliegue con Docker Compose

## Hadoop



Framework open-source para aplicaciones distribuidas que trabajan con Big Data

- Procesamiento de enormes cantidades de datos en grandes clusters de hardware barato (commodity clusters)
  - ▷ Escala: petabytes de datos en miles de nodos

## Arquitectura de Hadoop

### Tres partes

- Almacenamiento distribuido: HDFS
- Planificación de tareas y negociación de recursos: YARN
- Procesamiento distribuido: MapReduce

### Ventajas

- Bajo coste: clusters baratos o cloud
- Facilidad de uso
- Tolerancia a fallos

## Arquitectura de Hadoop

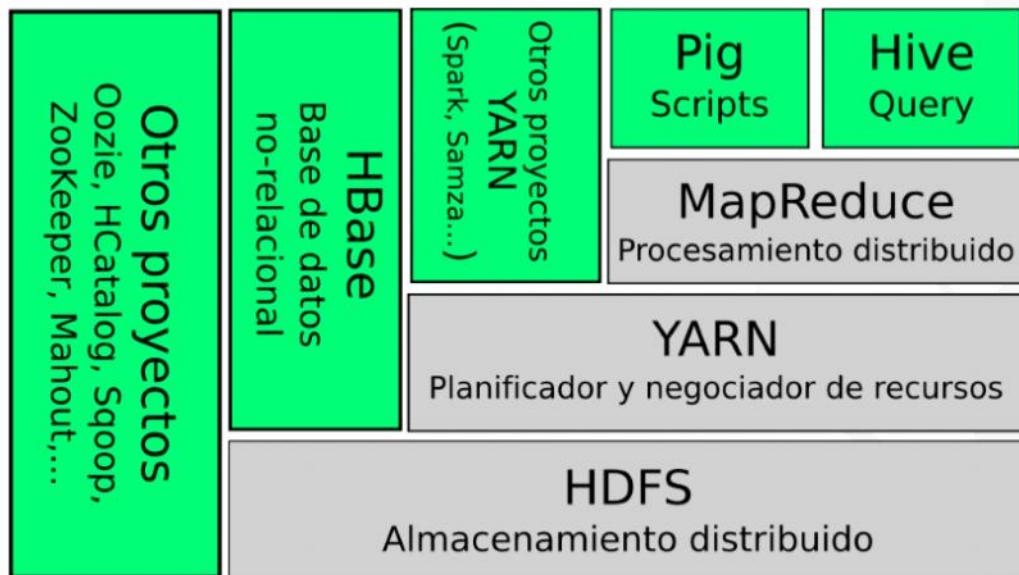
### Tres partes

- Almacenamiento distribuido: HDFS
- Planificación de tareas y negociación de recursos: YARN
- Procesamiento distribuido: MapReduce

### Ventajas

- Bajo coste: clusters baratos o cloud
- Facilidad de uso
- Tolerancia a fallos

## Componentes de Hadoop

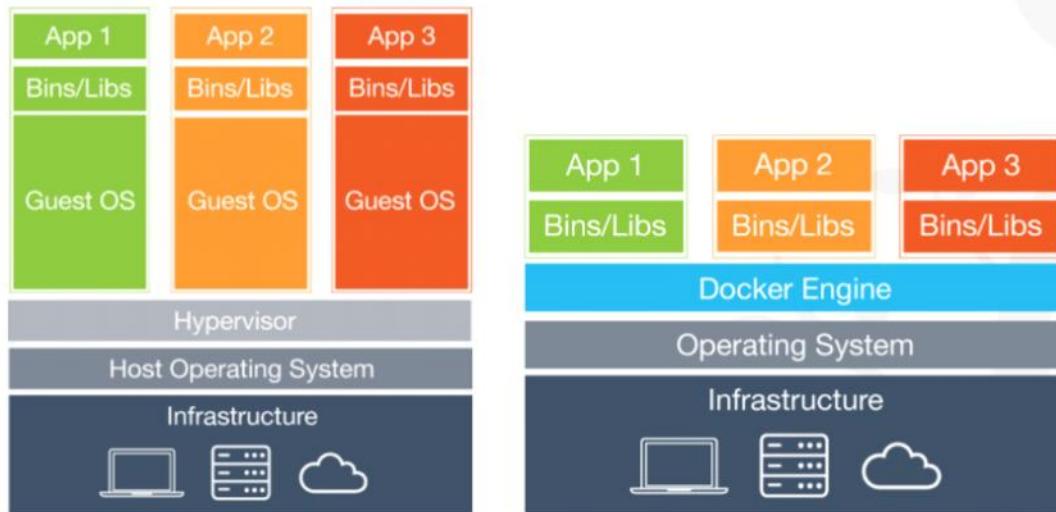


## Hands-on: Instalación base de Hadoop con Docker



## Conceptos de Docker

Usaremos contenedores Docker como máquinas virtuales con bajo overhead



## Conceptos de Docker

- **Docker Desktop:** aplicación para la gestión de contenedores
- **Imagen:** archivo que contiene todo lo necesario para ejecutar un contenedor
- **Volumen:** espacios de disco que pueden compartirse entre contenedores o entre un contenedor y el host
- **Redes:** permiten conectar contenedores que se ejecuten en un mismo o diferentes hosts
- **Dockerfile:** fichero de texto con instrucciones para construir una imagen Docker
- **Compose:** herramienta para facilitar el despliegue de múltiples contenedores que se ejecutan en el mismo host

## Hands-on: Instalación base de Hadoop con Docker

Pasos:

1. Inicia el Docker Desktop
2. Accede a la carpeta Hadoop\_cluster/Base/ y examina el contenido del fichero Dockerfile
3. Abre un terminal en esa carpeta y construye la imagen ejecutando

```
docker build -t hadoop-base-image .
```

▷ Una vez construida, puedes verla en el Docker Desktop o ejecutando:

```
docker image ls
```

## Índice

1 Componentes de Hadoop

2 HDFS: Hadoop Distributed File System

Despliegue del servicio NameNode

3 Gestión de recursos con YARN

Despliegue del servicio ResourceManager

4 Programación MapReduce

Despliegue de los DataNodes/NodeManagers

5 Automatización del despliegue con Docker Compose

## HDFS: Hadoop Distributed File System

### HDFS: Ventajas

- Diseñado para almacenar ficheros muy grandes en *commodity hardware*
- Elevado ancho de banda
- Fiabilidad mediante replicación
  - ▷ Los ficheros se dividen en bloques que se replican en diferentes nodos del cluster

### HDFS: Inconvenientes

- Elevada latencia
- Poco eficiente con muchos ficheros pequeños

## Demonios de HDFS

### Namenode

Mantiene la información (metadatos) de los ficheros y bloques que residen en el HDFS

### Datanodes

Mantienen los bloques de datos

- No tienen idea sobre los ficheros

## Otros conceptos HDFS

### Bloques

- Por defecto 128 MB, tamaño configurable
- Por defecto, cada bloque se replica tres veces (configurable)

### Checkpoint node

Mantienen checkpoints del Namenode

- debería ejecutarse en un sistema con características similares al Namenode

## HDFS: propiedades configurables (I)

Múltiples propiedades configurables (fichero `hdfs-site.xml`)

- `dfs.namenode.name.dir`: lista (separada por comas) de directorios donde el Namenode guarda sus metadatos (una copia en cada directorio), por defecto `file://${hadoop.tmp.dir}/dfs/name`
- `dfs.datanode.data.dir`: lista (separada por comas) de directorios donde los Datanodes guarda los bloques de datos (cada bloque en sólo uno de los directorios), por defecto `file://${hadoop.tmp.dir}/dfs/data`
- `dfs.namenode.checkpoint.dir`: lista (separada por comas) de directorios donde el CheckPoint node guarda los checkpoints (una copia en cada directorio), por defecto `file://${hadoop.tmp.dir}/dfs/namesecondary`

## HDFS: propiedades configurables (II)

- `dfs.blocksize`: tamaño de bloque para nuevos ficheros, por defecto 128MB
- `dfs.replication`: nº de réplicas por bloque, por defecto 3
- `dfs.replication.max`: máximo nº de réplicas permitido por bloque, por defecto 512
- `dfs.namenode.replication.min`: mínimo nº de réplicas permitido por bloque, por defecto 1

## Interfaz con HDFS

Varias interfaces:

1. Interfaz en línea de comandos: comando `hdfs dfs`
2. Interfaz web
3. Interfaz Java

Interfaz en línea de comandos:

- Permite cargar, descargar y acceder a los ficheros HDFS desde línea de comandos
- Ayuda: `hdfs dfs -help`

Más información:

[hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-hdfs/HDFSCCommands.html](http://hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-hdfs/HDFSCCommands.html),  
[hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-common/FileSystemShell.html](http://hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-common/FileSystemShell.html)

## Interfaz con HDFS

Varias interfaces:

1. Interfaz en línea de comandos: comando `hdfs dfs`
2. Interfaz web
3. Interfaz Java

Interfaz en línea de comandos:

- Permite cargar, descargar y acceder a los ficheros HDFS desde línea de comandos
- Ayuda: `hdfs dfs -help`

Más información:

[hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html](http://hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html),  
[hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-common/FileSystemShell.html](http://hadoop.apache.org/docs/stable3/hadoop-project-dist/hadoop-common/FileSystemShell.html)

## Interfaz en línea de comandos (I)

Algunos comandos de manejo de ficheros

| Comando   | Significado                         |
|---|-------------------------------------|
| <code>hdfs dfs -ls &lt;path&gt;</code>            | Lista ficheros                      |
| <code>hdfs dfs -ls -R &lt;path&gt;</code>         | Lista recursivamente                |
| <code>hdfs dfs -cp &lt;src&gt; &lt;dst&gt;</code> | Copia ficheros HDFS a HDFS          |
| <code>hdfs dfs -mv &lt;src&gt; &lt;dst&gt;</code> | Mueve ficheros HDFS a HDFS          |
| <code>hdfs dfs -rm &lt;path&gt;</code>            | Borra ficheros en HDFS              |
| <code>hdfs dfs -rm -r &lt;path&gt;</code>         | Borra recursivamente                |
| <code>hdfs dfs -cat &lt;path&gt;</code>           | Muestra fichero en HDFS             |
| <code>hdfs dfs -tail &lt;path&gt;</code>          | Muestra el final del fichero        |
| <code>hdfs dfs -stat &lt;path&gt;</code>          | Muestra estadísticas del fichero    |
| <code>hdfs dfs -mkdir &lt;path&gt;</code>         | Crea directorio en HDFS             |
| <code>hdfs dfs -chmod ...</code>                  | Cambia permisos de fichero          |
| <code>hdfs dfs -chown ...</code>                  | Cambia propietario/grupo de fichero |

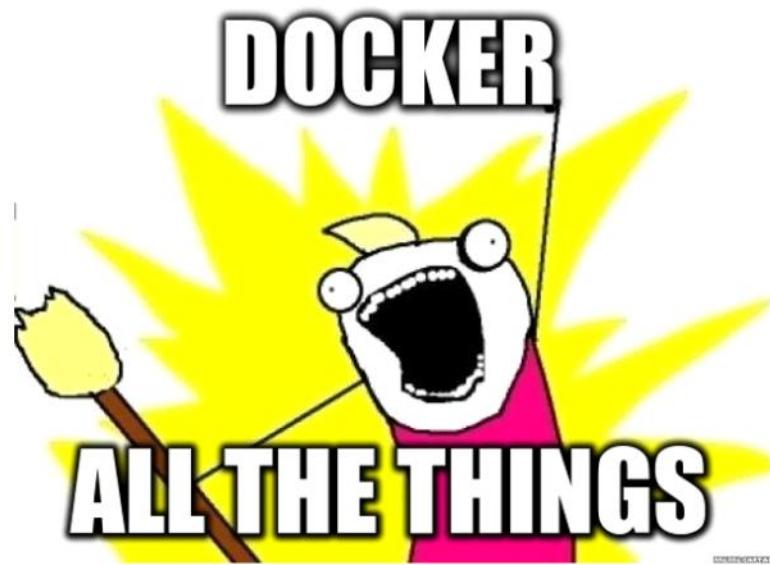
## Interfaz en línea de comandos (II)

Movimiento de ficheros del sistema local al HDFS:

| Comando                     | Significado           |
|-----------------------------|-----------------------|
| hdfs dfs -put <local> <dst> | Copia de local a HDFS |
| hdfs dfs -copyFromLocal ... | Igual que -put        |
| hdfs dfs -moveFromLocal ... | Mueve de local a HDFS |
| hdfs dfs -get <src> <loc>   | Copia de HDFS a local |
| hdfs dfs -copyToLocal ...   | Copia de HDFS a local |

Más información: <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html>

## Hands-on: Instalación del servidor NameNode



## Hands-on: Creación de la imagen para el NameNode

Pasos:

1. Accede a la carpeta Hadoop\_cluster/NameNode/ y examina el contenido del fichero Dockerfile
2. Examina también los ficheros dentro de la carpeta Hadoop\_cluster/NameNode/Config-files
3. Abre un terminal en la carpeta Hadoop\_cluster/NameNode/ y construye la imagen del NameNode ejecutando

```
docker build -t namenode-image .
```

▷ Una vez construida, puedes verla en el Docker Desktop o ejecutando:

```
docker image ls
```

## Hands-on: Ejecución del servidor NameNode

Pasos:

1. Crea la red que conectará los contenedores ejecutando:

```
docker network create hadoop-net
```

▷ Puedes ver la red creada con: docker network inspect hadoop-net

2. Inicia un contenedor corriendo el servicio NameNode con:

```
docker container run --rm --init --detach --name namenode  
--network=hadoop-net --hostname namenode -p 9870:9870 namenode-image
```

## Hands-on: Ejecución del servidor NameNode

3. Accede al interfaz web del NameNode en <http://localhost:9870>

4. Accede a un shell en el NameNode ejecutando en el terminal:

```
docker container exec -ti namenode /bin/bash
```

- ▷ Prueba el comando hdfs
- ▷ Sal del NameNode usando exit

## Hands-on: Ejecución del servidor NameNode

3. Accede al interfaz web del NameNode en <http://localhost:9870>

4. Accede a un shell en el NameNode ejecutando en el terminal:

```
docker container exec -ti namenode /bin/bash
```

- ▷ Prueba el comando hdfs
- ▷ Sal del NameNode usando exit

## Índice

1 Componentes de Hadoop

2 HDFS: Hadoop Distributed File System  
Despliegue del servicio NameNode

3 Gestión de recursos con YARN  
Despliegue del servicio ResourceManager

4 Programación MapReduce  
Despliegue de los DataNodes/NodeManagers

5 Automatización del despliegue con Docker Compose

## YARN: Yet Another Resource Negotiator

Se encarga de la gestión de recursos y job-scheduling/monitorización usando tres demonios:

- **ResourceManager (RM)**: planificador general
- **NodeManagers (NM)**: monitorización, uno por nodo
- **ApplicationMasters (AM)**: gestión de aplicaciones, uno por aplicación

Permite que diferentes tipos de aplicaciones (no solo MapReduce) se ejecuten en el cluster

- Las aplicaciones se despliegan en “contenedores”, que son máquinas virtuales de Java (JVMs)

## YARN: Yet Another Resource Negotiator

Se encarga de la gestión de recursos y job-scheduling/monitorización usando tres demonios:

- *ResourceManager (RM)*: planificador general
- *NodeManagers (NM)*: monitorización, uno por nodo
- *ApplicationMasters (AM)*: gestión de aplicaciones, uno por aplicación

Permite que diferentes tipos de aplicaciones (no solo MapReduce) se ejecuten en el cluster

- Las aplicaciones se despliegan en “contenedores”, que son máquinas virtuales de Java (JVMs)

## Demonios YARN (I)

### ResourceManager

- arbitra los recursos entre las aplicaciones en el sistema
- demonio global, obtiene datos del estado del cluster de los node managers

### NodeManagers

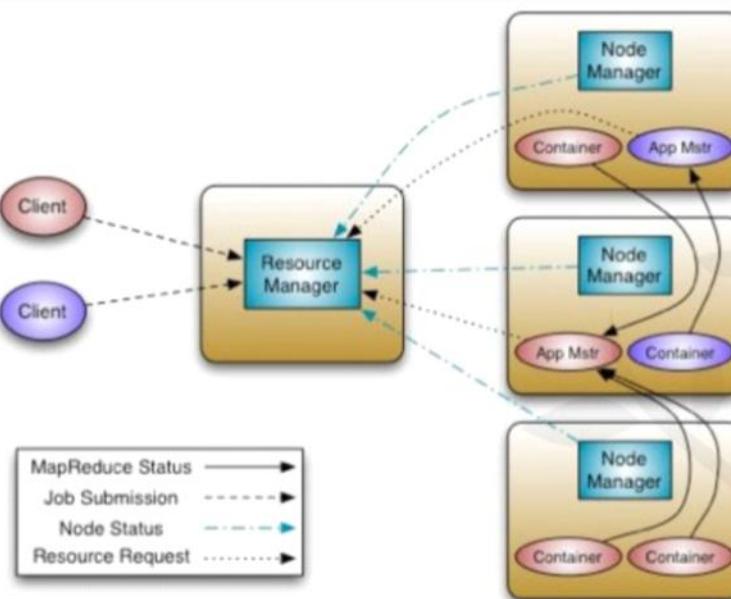
- uno por nodo
- monitorizan los recursos del cluster

## Demonios YARN (II)

### ApplicationMasters

- uno por aplicación, se encarga de gestionar el ciclo de vida de la aplicación
- solicita recursos (**contenedores**) al ResourceManager y ejecuta la aplicación en esos contenedores
  - ▷ el AM se ejecuta en su propio contenedor
- trabaja con los Nodemangers para ejecutar y monitorizar las tareas

## Elementos de control YARN



Fuente: A. Murthy, V. Vavilapalli, "Apache Hadoop YARN", Addison-Wesley, marzo 2014

## YARN: propiedades configurables (I)

Múltiples propiedades configurables (fichero `yarn-site.xml`)

- `yarn.resourcemanager.hostname`: el host ejecutando el ResourceManager
- `yarn.nodemanager.aux-services`: lista de servicios auxiliares que deben implementar los NodeManagers (uno de ellos, el barajado MapReduce)
- `yarn.nodemanager.resource.memory-mb`: cantidad de memoria que puede reservarse para contenedores YARN en un nodo

## YARN: propiedades configurables (II)

- `yarn.scheduler.maximum-allocation-vcores`,  
`yarn.scheduler.minimum-allocation-vcores`: nº máximo y mínimo de cores virtuales (threads) que pueden ser concedidos a un contenedor
- `yarn.scheduler.maximum-allocation-mb`,  
`yarn.scheduler.minimum-allocation-mb`: memoria máxima y mínima que puede ser concedida a un contenedor (la memoria solicitada se redondea a un múltiplo del mínimo)

## Comando yarn

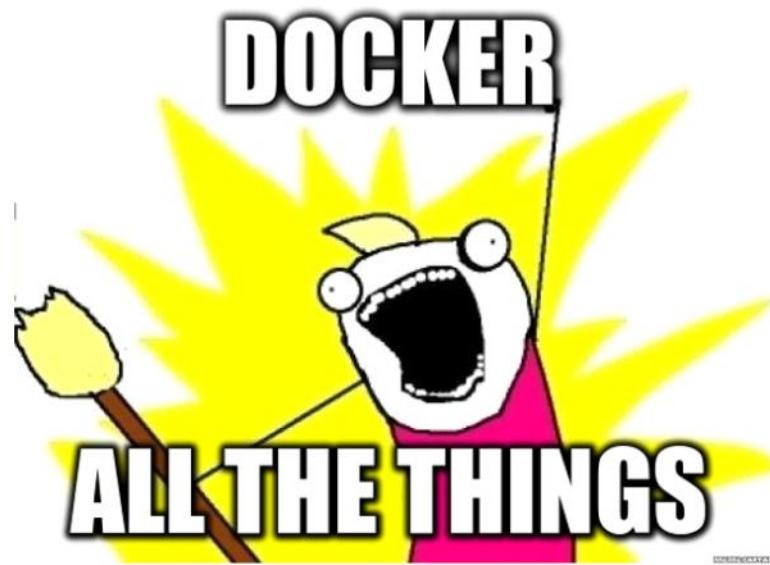
Permite lanzar y gestionar trabajos en YARN:

- `yarn jar`: ejecuta un fichero jar
- `yarn application`: información sobre las aplicaciones ejecutándose en YARN
- `yarn container`: información sobre los contenedores
- `yarn node`: información sobre los nodos
- `yarn top`: información sobre el uso del cluster
- `yarn rmadmin`: comandos para la administración del cluster

Más información:

[hadoop.apache.org/docs/stable3/hadoop-yarn/hadoop-yarn-site/YarnCommands.html](http://hadoop.apache.org/docs/stable3/hadoop-yarn/hadoop-yarn-site/YarnCommands.html)

## Hands-on: Instalación del servidor ResourceManager



## Hands-on: Creación de la imagen para el ResourceManager

Pasos:

1. Accede a la carpeta Hadoop\_cluster/ResourceManager/ y examina el contenido del fichero Dockerfile
  2. Examina también los ficheros dentro de la carpeta Hadoop\_cluster/ResourceManager/Config-files
  3. Abre un terminal en la carpeta Hadoop\_cluster/ResourceManager/ y construye la imagen del ResourceManager ejecutando
- ```
docker build -t resourcemanager-image .
```
- ▷ Una vez construida, puedes verla en el Docker Desktop o ejecutando:

```
docker image ls
```

## Hands-on: Ejecución del servidor ResourceManager

Pasos:

1. Inicia un contenedor corriendo el servicio ResourceManager con:

```
docker container run --rm --init --detach --name resourcemanager  
--network=hadoop-net --hostname resourcemanager -p 8088:8088  
resourcemanager-image
```

2. Accede al interfaz web del ResourceManager en <http://localhost:8088>

## Índice

- 1 Componentes de Hadoop**
- 2 HDFS: Hadoop Distributed File System**  
Despliegue del servicio NameNode
- 3 Gestión de recursos con YARN**  
Despliegue del servicio ResourceManager
- 4 Programación MapReduce**  
Despliegue de los DataNodes/NodeManagers
- 5 Automatización del despliegue con Docker Compose**

## Programación MapReduce

Modelo de programación data-parallel diseñado para escalabilidad y tolerancia a fallos en grandes sistemas de commodity hardware

- Basado en la combinación de operaciones Map y Reduce
- Los datos de entrada se organizan como listas de pares clave/valor
  - ▷ Esos datos están distribuidos y replicados en HDFS
- Map: aplica la misma función a diferentes bloques de entrada
  - ▷ Operación totalmente paralela: las funciones map se ejecutan sobre los nodos del cluster
  - ▷ Cada función map lee los datos del nodo en el que se ejecuta
- Reduce: realiza una operación de reducción sobre las salidas de los map
  - ▷ Puede haber varios reduces

## Ejemplo: WordCount

Cuenta las ocurrencias de cada palabra en ficheros de texto

- **Entrada:**

$$K_1 = \emptyset, V_1 = \text{línea}$$

- **Salida:**

*pares (palabra, nº de ocurrencias)*

- Pseudocódigo del Map

```
map(key, value) :
    //key: nada, value: línea de texto
    for each word w in value emit(w, 1)
```

- Pseudocódigo del Reduce

```
reduce(key, values) :
    //key: palabra; values: un iterador sobre los ls
    emit(key, sum(values))
```

## Ejemplo: WordCount

Cuenta las ocurrencias de cada palabra en ficheros de texto

- **Entrada:**

$$K_1 = \emptyset, V_1 = \text{línea}$$

- **Salida:**

*pares (palabra, nº de ocurrencias)*

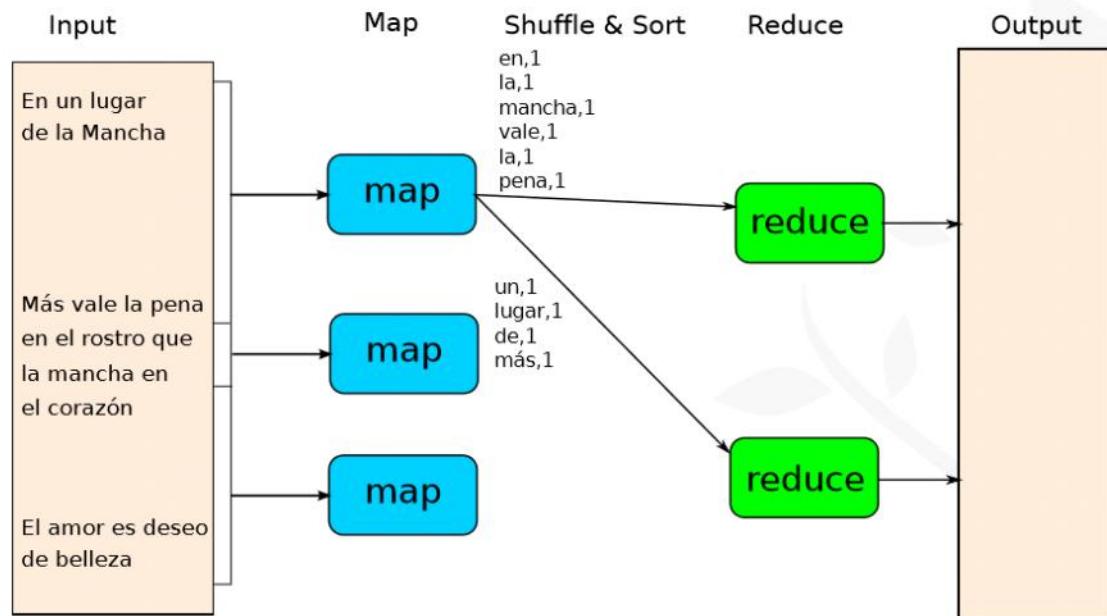
- Pseudocódigo del Map

```
map(key, value) :
    //key: nada, value: línea de texto
    for each word w in value emit(w, 1)
```

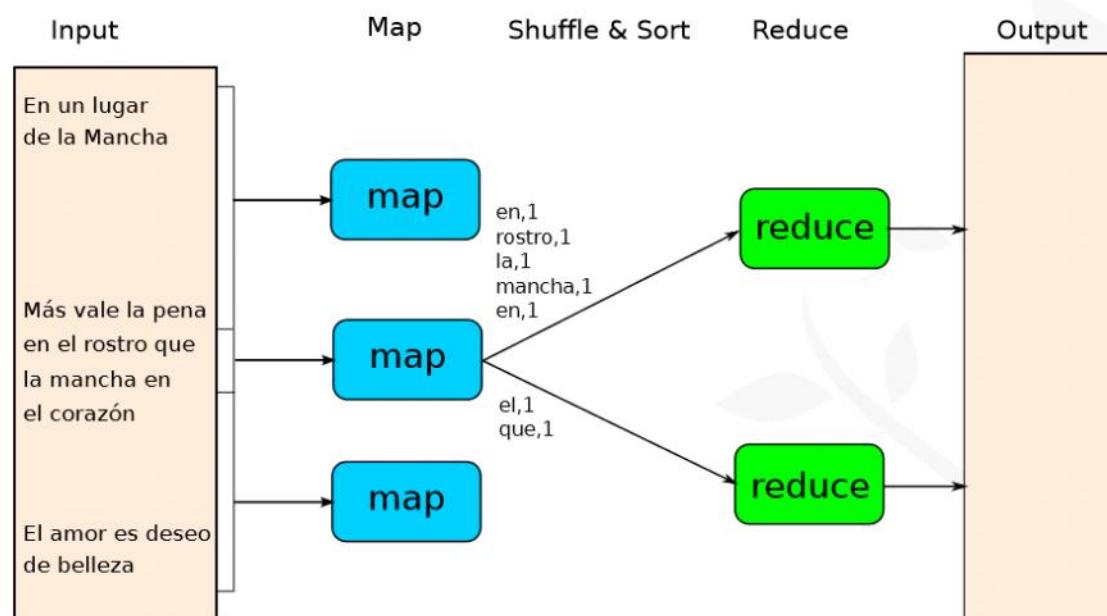
- Pseudocódigo del Reduce

```
reduce(key, values) :
    //key: palabra; values: un iterador sobre los ls
    emit(key, sum(values))
```

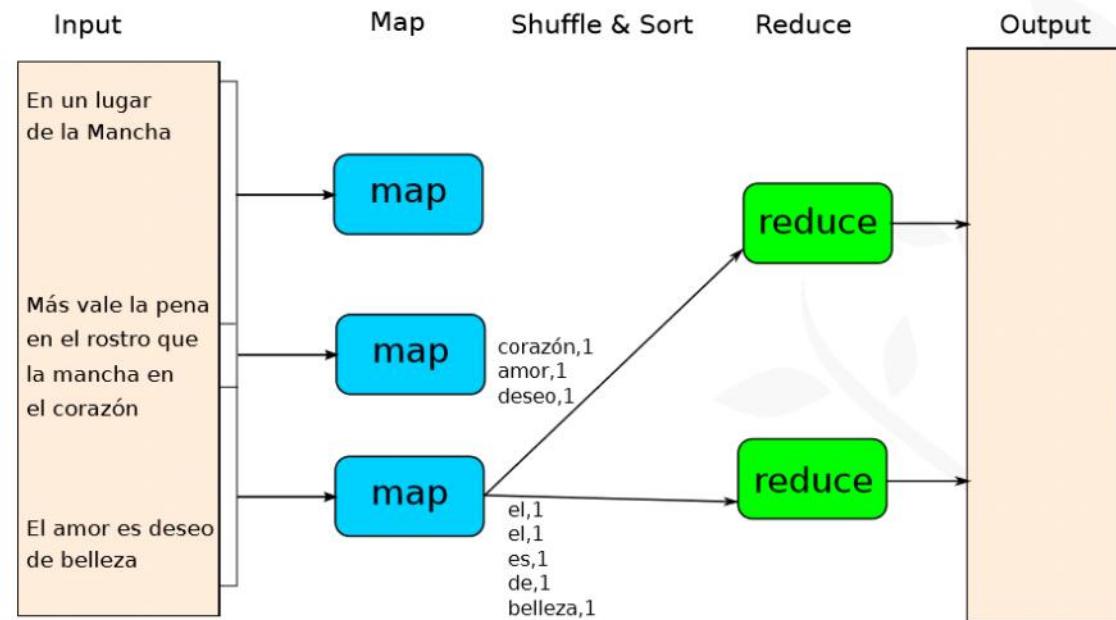
## Ejemplo: WordCount



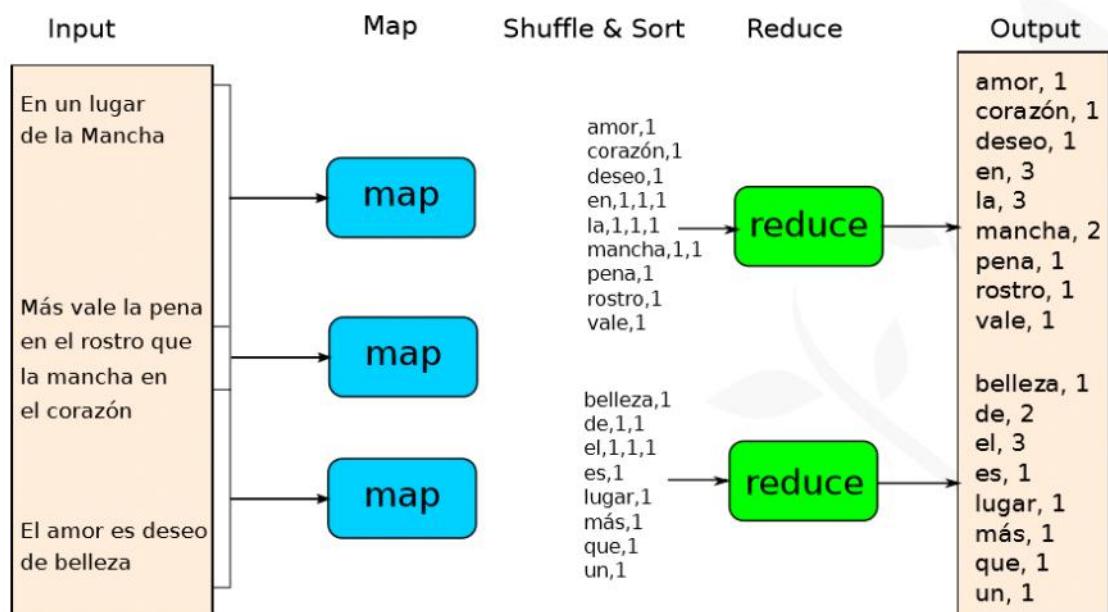
## Ejemplo: WordCount



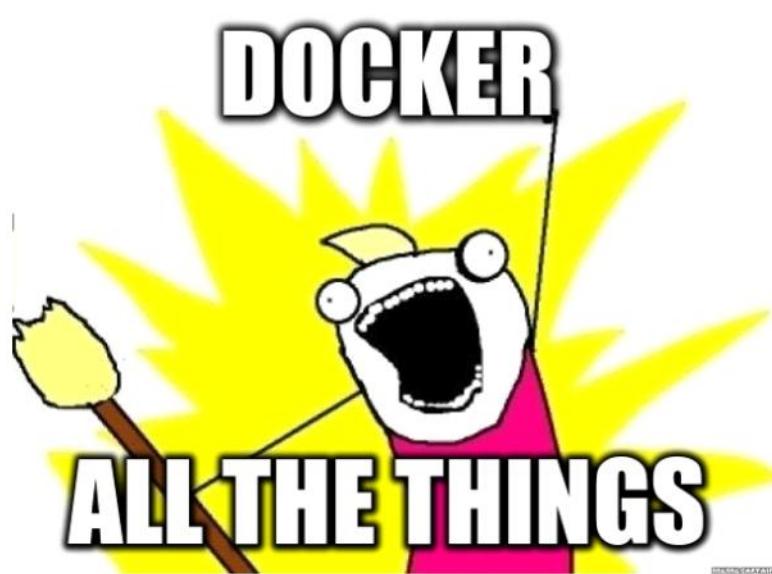
## Ejemplo: WordCount



## Ejemplo: WordCount



## Hands-on: Instalación de los DataNodes/NodeManagers



## Hands-on: Creación de la imagen para los DataNodes/NodeManagers

Pasos:

1. Accede a la carpeta Hadoop\_cluster/DataNode–NodeManager/ y examina el contenido del fichero Dockerfile
2. Examina también los ficheros dentro de la carpeta Hadoop\_cluster/DataNode–NodeManager/Config–files
3. Abre un terminal en la carpeta Hadoop\_cluster/DataNode–NodeManager/ y construye la imagen ejecutando

```
docker build -t dnmm-image .
```

▷ Una vez construida, puedes verla en el Docker Desktop o ejecutando:

```
docker image ls
```

## Hands-on: Ejecución de los DataNodes/NodeManagers

Pasos:

1. Inicia cuatro contenedores corriendo los servicios DataNode/NodeManager con:

```
docker container run --rm --init --detach --name dnnm1
--network=hadoop-net --hostname dnnm1 dnnm-image
docker container run --rm --init --detach --name dnnm2
--network=hadoop-net --hostname dnnm2 dnnm-image
docker container run --rm --init --detach --name dnnm3
--network=hadoop-net --hostname dnnm3 dnnm-image
docker container run --rm --init --detach --name dnnm4
--network=hadoop-net --hostname dnnm4 dnnm-image
```

2. Accede a los interfaces web del NameNode y del ResourceManager y comprueba que se han registrado los 4 workers

## Hands-on: Prueba del HDFS

1. Accede a un shell en el NameNode ejecutando en el terminal:

```
docker container exec -ti namenode /bin/bash
```

2. Crea un fichero grande usando:

```
dd if=/dev/urandom of=fich300M bs=1M count=300
```

3. Copia al fichero al HDFS y bórralo de local

```
hdfs dfs -put fich300M /user/hdadmin; rm fich300M
```

4. En el interfaz web del NameNode ve a Utilities → Browse the filesystem y obtén la información del fichero

- ▷ ¿En cuántos bloques se ha dividido?
- ▷ ¿Dónde residen esos bloques?

5. Sal del NameNode ejecutando exit

## Hands-on: Ejecución de una aplicación MapReduce con YARN

1. Accede a un shell en el ResourceManager ejecutando en el terminal:

```
docker container exec -ti resourcemanager /bin/bash
```

2. Ejecuta una aplicación MR de ejemplo que aproxima el valor de  $\pi$ :

```
export MR_EXAMPLES=$HADOOP_HOME/share/hadoop/mapreduce  
yarn jar $MR_EXAMPLES/hadoop-mapreduce-examples-*.jar pi 16 1000
```

3. En el interfaz web del ResourceManager comprueba que la aplicación aparece en estado FINISHED

- ▷ ¿En qué nodo se ejecutó el Application Master?
- ▷ ¿Cuántos contenedores MR se ejecutaron?

## Índice

- 1 Componentes de Hadoop
- 2 HDFS: Hadoop Distributed File System  
Despliegue del servicio NameNode
- 3 Gestión de recursos con YARN  
Despliegue del servicio ResourceManager
- 4 Programación MapReduce  
Despliegue de los DataNodes/NodeManagers
- 5 Automatización del despliegue con Docker Compose

## Hands-on: Automatización del despliegue con Docker Compose



## Hands-on: Automatización del despliegue con Docker Compose

Levantar todos los servicios “a mano” puede ser complejo y propenso a errores

- Compose permite automatizar el despliegue de un stack de servicios complejo

Para ejecutar un conjunto de contenedores con compose, se siguen los siguientes pasos:

- Se describen los contenedores en un archivo YAML llamado `docker-compose.yml`
- Se usa el comando `docker compose` para construir las imágenes (si es necesario) e iniciar los servicios

## Hands-on: Automatización del despliegue con Docker Compose

Pasos:

1. Detén los contenedores desde el Docker Desktop o ejecutando en el terminal:

```
docker container stop dnnm1 dnnm2 dnnm3 dnnm4  
docker container stop namenode resourcemanager
```

▷ Todos los contenedores se paran y desaparecen

2. Examina el fichero docker-compose.yml

3. Levanta el cluster con 4 DataNodes/NodeManagers ejecutando:

```
docker compose up --scale dnnm=4 -d
```

▷ Comprueba que todo se ha iniciado correctamente

▷ Puedes parar todos los contenedores con docker compose stop y reiniciarlos con docker compose start

▷ Para eliminar los contenedores, usa docker compose down

# Deploy with Docker

domingo, 12 de noviembre de 2023 16:34

[Curso Gratuito de Hadoop: "Despliegue de un cluster Hadoop mediante contenedores Docker Big Data"](#)



Instalar docker-desktop

<https://www.docker.com/products/docker-desktop/>

Descargarse los archivos y comentarlos

<https://recursos.universidadviu.es/int-guia-gratuita-hado>



Hadoop\_co  
n\_Docker  
op



VIU-Hadoo  
p\_con\_D...

Partes sueltas del código:

```
docker build -t hadoop-base-image .
```

```
docker container run --rm --init --detach --name namenode --network=hadoop-net --hostname  
namenode -p 9870:9870 namenode-image
```

```
docker container run --rm --init --detach --name resourcemanager --network=hadoop-net --  
hostname resourcemanager -p 8088:8088 resourcemanager-image
```

```
docker container run --rm --init --detach --name dnnm1 --network=hadoop-net --hostname dnnm1  
dnnm-image
```

```
docker container run --rm --init --detach --name dnnm2 --network=hadoop-net --hostname dnnm2  
dnnm-image
```

```
docker container run --rm --init --detach --name dnnm3 --network=hadoop-net --hostname dnnm3  
dnnm-image  
docker container run --rm --init --detach --name dnnm4 --network=hadoop-net --hostname dnnm4  
dnnm-image
```

```
export MR_EXAMPLES=$HADOOP_HOME/share/hadoop/mapreduce  
yarn jar $MR_EXAMPLES/hadoop-mapreduce-examples-*.jar pi 16 1000
```

```
docker container stop dnnm1 dnnm2 dnnm3 dnnm4  
docker container stop namenode resourcemanager
```

```
docker compose
```

# CLUSTER WITH RaspberryPIs

domingo, 29 de mayo de 2022 16:41

<https://towardsdatascience.com/assembling-a-personal-data-science-big-data-laboratory-in-a-raspberry-pi-4-or-vms-cluster-e4c5a0473025>

# ETL Pipeline with Apache Spark

miércoles, 11 de diciembre de 2024 2:12

The screenshot shows a Medium article card. At the top left, it says "10 min read". The main title is "Creating a Simple ETL Pipeline With Apache Spark" in large, bold, black font. Below the title is a subtitle: "Transforming raw data into a star schema with simple (py)spark code". To the right of the title, the author's name "joaopedro214.medium.com" is visible. At the bottom left, there is a small circular profile picture of the author, João Pedro, and his name next to it. On the far right, the Medium logo is present.

10 min read

## Creating a Simple ETL Pipeline With Apache Spark

Transforming raw data into a star schema with simple (py)spark code

joaopedro214.medium.com

João Pedro

Medium

# Spark Analysis

lunes, 30 de mayo de 2022 20:31

[Using Apache Spark 2.0 to Analyze the City of San Francisco's Open Data](#)



# Steps to create a Hadoop and Spark Cluster

jueves, 12 de diciembre de 2024 1:33

Que hay que tener en cuenta para instalar un cluster de hadoop y spark de un Master y 2 nodos esclavos. Revisa cada punto importante, desde el hardware a utilizar (RAM, CPU, arquitectura de procesador, Sistema Operativo, redes) hasta el software necesario para correrlo

Voy a desglosar detalladamente los aspectos fundamentales para instalar un cluster de Hadoop y Spark con un nodo maestro y dos nodos esclavos:

## 1. Requisitos de Hardware

### A. Nodo Maestro:

- CPU: Procesador Intel Xeon o AMD EPYC
  - \* Mínimo 8 núcleos
  - \* Recomendado 16-24 núcleos
  - \* Arquitectura x86\_64
- RAM:
  - \* Mínimo 32 GB
  - \* Recomendado 64-128 GB
- Almacenamiento:
  - \* SSD NVMe
  - \* 500 GB - 2 TB
  - \* RAID opcional para mayor redundancia

### B. Nodos Esclavos (2 nodos):

- CPU: Similar al maestro
  - \* 8-16 núcleos
  - \* Procesadores con soporte de virtualización
- RAM:
  - \* Mínimo 32 GB
  - \* Recomendado 64 GB
- Almacenamiento:
  - \* SSD o HDD de alta velocidad
  - \* 1-4 TB por nodo
  - \* Configuración JBOD o RAID según necesidad

## 2. Requisitos de Red

- Switch de red dedicado
- Conexiones 10 Gigabit Ethernet
- Latencia baja (<1 ms entre nodos)
- Ancho de banda mínimo 1 Gbps
- Red privada con configuración:
  - \* IP estáticas
  - \* Resolución de nombres DNS
  - \* Firewall configurado
  - \* Reglas de comunicación entre nodos

### 3. Sistema Operativo

Distribuciones recomendadas:

- CentOS/RHEL 8.x
- Ubuntu Server 20.04 LTS
- Rocky Linux

Características:

- Versión de 64 bits
- Kernel Linux reciente
- Soporte a largo plazo
- Actualizaciones de seguridad
- Compatibilidad con contenedores

### 4. Preparación de Sistemas

Configuraciones previas:

- Sincronización de hora (NTP)
- Usuarios y grupos dedicados
- Configuración de SSH sin contraseña
- Desactivar SELinux/AppArmor
- Configurar swappiness bajo
- Optimizar parámetros del kernel

### 5. Software Base

Instalaciones necesarias:

- Java Development Kit (JDK)
  - \* Versión 8 o 11
  - \* OpenJDK recomendado
- Python 3.7+
- Herramientas de red
  - \* iproute2
  - \* net-tools
- Compiladores
  - \* gcc
  - \* g++

### 6. Componentes de Big Data

Instalaciones principales:

- Hadoop
  - \* Versión 3.x
  - \* Distribución Apache o CDH
- Spark
  - \* Versión 3.x
- HDFS
- YARN
- MapReduce

### 7. Configuraciones Adicionales

- Configurar /etc/hosts
- Generar claves SSH
- Definir variables de entorno
- Configurar archivos core-site.xml
- Definir topología de red
- Configurar políticas de seguridad

## 8. Herramientas de Monitoreo

Recomendadas:

- Ganglia
- Prometheus
- Grafana
- Nagios
- Zabbix

## 9. Consideraciones de Seguridad

- Kerberos
- Apache Ranger
- Cifrado en reposo y tránsito
- Control de acceso
- Auditoría de accesos

## 10. Consideraciones de Alto Rendimiento

- Configurar storage con alta velocidad
- Optimizar configuraciones de red
- Balanceo de carga
- Tolerancia a fallos
- Réplicas de datos

## 11. Alternativas de Distribución

- Cloudera
- Hortonworks
- MapR
- Amazon EMR
- Azure HDInsight

Pasos de Implementación:

1. Preparar infraestructura
2. Instalar sistema operativo
3. Configurar red
4. Instalar Java
5. Configurar usuarios
6. Instalar Hadoop/Spark
7. Configurar archivos
8. Iniciar servicios
9. Validar cluster
10. Implementar monitoreo

Recomendación Final:

- Realizar pruebas incrementales
- Documentar cada configuración
- Mantener actualizaciones
- Implementar respaldos
- Planificar escalabilidad

Esta guía proporciona una visión integral para implementar un cluster Hadoop/Spark robusto y eficiente. La implementación específica dependerá de tus requisitos particulares de procesamiento de datos.

¿Necesitas que profundice en alguno de estos puntos o tienes alguna duda sobre la implementación?

Lo podemos hacer con 3 raspberry pi'S?

Sí, puedes crear un cluster Hadoop y Spark con Raspberry Pi, pero hay algunas consideraciones importantes:

Ventajas:

- Bajo costo
- Consumo energético reducido
- Ideal para proyectos educativos y experimentales

Limitaciones de Hardware Raspberry Pi 4:

- CPU: ARM Cortex-A72 (4 núcleos)
- RAM:
  - \* Modelo básico: 4 GB
  - \* Modelo avanzado: 8 GB
- Almacenamiento: MicroSD o SSD externo
- Red: Ethernet Gigabit

Configuración Recomendada:

1. Sistema Operativo

- Raspberry Pi OS (basado en Debian)
- 64 bits
- Última versión estable

2. Versiones de Software

- Java: OpenJDK 11
- Hadoop: Versión ligera 3.2.x
- Spark: Versión 3.x para ARM
- Python 3.7+

3. Configuración de Red

- Un Raspberry Pi como nodo maestro
- Dos Raspberry Pi como nodos esclavos
- Switch de red pequeño
- Conexión por cable Ethernet

4. Consideraciones Adicionales

- Refrigeración adecuada
- Fuente de alimentación estable
- Almacenamiento externo (SSD USB)

Pasos de Instalación:

1. Instalar sistema operativo
2. Configurar red
3. Instalar Java
4. Configurar SSH
5. Instalar Hadoop/Spark
6. Configurar cluster

Limitaciones de Rendimiento:

- Procesamiento más lento

- Menos memoria
- Ancho de banda limitado
- No apto para big data a gran escala

Recomendado para:

- Aprendizaje
- Proyectos pequeños
- Experimentación
- Desarrollo educativo

Enumérame las opciones de instalación de un cluster que tengo:

En línea de comando en máquina local

En línea de comando en resto de máquinas

Utilizar bash scripting para realizar los pasos anteriores más rápido.

A través de docker

A través de alguna plataforma de Cloud computing

Alguna distribución que ya lo lleve casi todo incluido e instalarla en una máquina virtual?

Esos son los que conozco.

Ponme en formato tabla todas las opciones que tengo valorando sus ventajas e inconvenientes.

Aquí tienes una tabla comparativa detallada de las opciones de instalación de un cluster de Hadoop/Spark:

Método	Ventajas	Inconvenientes	Complejidad	Tiempo de Implementación	Escalabilidad
Mejor Para					
Instalación Manual en Línea de Comando (Máquina Local)	- Control total - Aprendizaje profundo - Personalización completa	- Mayor tiempo de configuración - Propenso a errores - Requiere conocimiento técnico detallado	Alta	1-2 días	Baja
Investigadores   Educación					Desarrolladores
Instalación Manual en Múltiples Máquinas	- Configuración personalizada - Control detallado - Ideal para infraestructura propia	- Complejo de gestionar - Requiere configuración individual - Sincronización manual	Muy Alta	2-3 días	Media
					Centros de datos Grandes empresas Infraestructura propia
Bash Scripting	- Automatización - Reproducibilidad	- Menos errores humanos - Requiere mantenimiento de scripts	Media	1 día	Media
					DevOps Equipos de infraestructura Proyectos repetitivos
Docker	- Portabilidad - Aislamiento - Fácil replicación	- Gestión de dependencias - Overhead de contenedores	Media	4-8 horas	Alta
					Microservicios Desarrollo ágil Entornos de pruebas
Cloud Computing	- Escalabilidad infinita - Pago por uso	- Gestión automática - Dependencia del proveedor	Alta disponibilidad	2-4 horas	Baja
					Configuración inicial Empresas Proyectos grandes Producción
Distribuciones Preconfiguradas (Máquina Virtual)	- Instalación rápida	- Configuración lista - Menor complejidad técnica	Prototipado rápido	1-2 horas	Baja
					Pequeñas organizaciones

Opciones Específicas por Categoría:

#### 1. Instalación Manual

- Apache Hadoop + Apache Spark
- Configuración desde cero

- Mayor curva de aprendizaje

## 2. Docker

- Imagen oficial Apache
- docker-compose para cluster
- Imágenes de Cloudera
- Bitnami Hadoop/Spark

## 3. Cloud Computing

- Amazon EMR
- Google Dataproc
- Azure HDInsight
- Databricks
- Oracle Cloud

## 4. Distribuciones Preconfiguradas

- Cloudera CDH
- Hortonworks HDP
- MapR Distribution
- Apache BigTop
- Pivotal HD

### Recomendación Personal:

- Principiantes: Docker o Distribución Preconfigurada
- Desarrollo: Instalación Manual o Bash Scripting
- Producción: Cloud Computing

### Consideraciones Adicionales:

- Evalúa tus recursos computacionales
- Considera la curva de aprendizaje
- Analiza requisitos específicos del proyecto
- Piensa en la escalabilidad futura

### Pasos para Elegir:

1. Define tu caso de uso
2. Evalúa recursos disponibles
3. Considera presupuesto
4. Valora complejidad técnica
5. Proyecta crecimiento futuro

¿Quieres que profundice en alguna de estas opciones o tienes un caso de uso específico?

**Podrías listarme los pasos a seguir para instalar el cluster de hadoop y spark en ubuntu. Con todos los comandos de terminal y los pasos que hay que seguir.**

Claro, te proporcionaré una guía completa paso a paso para instalar un cluster de Hadoop y Spark en Ubuntu. Asumiré una configuración con un nodo maestro y dos nodos esclavos.

### PREPARACIÓN PREVIA

1. Actualizar todos los nodos

```
```bash
```

```
sudo apt update && sudo apt upgrade -y  
```
```

## 2. Instalar paquetes básicos

```
```bash
sudo apt install -y openjdk-11-jdk ssh pdsh curl
```
```

## 3. Configurar resolución de nombres

Editar `/etc/hosts` en TODOS los nodos:

```
```bash
sudo nano /etc/hosts
```
```

Añadir algo como:

```
```bash
192.168.1.100 master
192.168.1.101 slave1
192.168.1.102 slave2
```
```

## 4. Crear usuario hadoop en TODOS los nodos

```
```bash
sudo addgroup hadoop
sudo adduser --ingroup hadoop hdadmin
sudo usermod -aG sudo hdadmin
```
```

## 5. Configurar SSH sin contraseña

En el nodo maestro:

```
```bash
su - hdadmin
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```
```

# Copiar clave a otros nodos (sustituir IP)

```
ssh-copy-id hdadmin@slave1
ssh-copy-id hdadmin@slave2
```
```

## 6. Descargar e instalar Hadoop

```
```bash
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.4/hadoop-3.3.4.tar.gz
tar -xvf hadoop-3.3.4.tar.gz
sudo mv hadoop-3.3.4 /opt/hadoop
```
```

## 7. Configurar variables de entorno

Editar `~/.bashrc`:

```
```bash
nano ~/.bashrc
```
```

Añadir:

```
```bash
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
```
```

## 8. Configuraciones de Hadoop

Editar archivos en `/opt/hadoop/etc/hadoop/`:

`core-site.xml`:

```
```xml
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
````
```

`hdfs-site.xml`:

```
```xml
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/opt/hadoop/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/opt/hadoop/data/datanode</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
</configuration>
````
```

`yarn-site.xml`:

```
```xml
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>master</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
````
```

`mapred-site.xml`:

```
```xml
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
````
```

`workers`:

```
```
```

```
slave1  
slave2  
```
```

#### 9. Crear directorios de datos

```
```bash  
sudo mkdir -p /opt/hadoop/data/namenode  
sudo mkdir -p /opt/hadoop/data/datanode  
sudo chown -R hdadmin:hadoop /opt/hadoop  
```
```

#### 10. Instalar Spark

```
```bash  
wget https://downloads.apache.org/spark/spark-3.3.2/spark-3.3.2-bin-hadoop3.tgz  
tar -xvf spark-3.3.2-bin-hadoop3.tgz  
sudo mv spark-3.3.2-bin-hadoop3 /opt/spark  
```
```

#### 11. Configurar Spark

Editar `spark-env.sh`:

```
```bash  
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64  
export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop  
export SPARK_HOME=/opt/spark  
```
```

#### 12. Formato de HDFS (SOLO EN NODO MASTER)

```
```bash  
hdfs namenode -format  
```
```

#### 13. Iniciar servicios (EN NODO MASTER)

```
```bash  
start-dfs.sh  
start-yarn.sh  
```
```

#### 14. Verificar servicios

```
```bash  
jps # Debe mostrar servicios de Hadoop/Yarn  
hdfs dfsadmin -report  
```
```

#### 15. Copiar configuraciones a otros nodos

```
```bash  
# Desde el nodo master  
scp -r /opt/hadoop/etc/hadoop/* slave1:/opt/hadoop/etc/hadoop/  
scp -r /opt/hadoop/etc/hadoop/* slave2:/opt/hadoop/etc/hadoop/  
```
```

### PASOS FINALES DE VERIFICACIÓN

#### 1. Test de HDFS

```
```bash  
hdfs dfs -mkdir /test  
hdfs dfs -put some_file /test  
hdfs dfs -ls /test
```

...

## 2. Test de MapReduce

```
```bash
yarn jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar pi 10
100
````
```

### CONSIDERACIONES IMPORTANTES:

- Estos pasos son para Ubuntu Server
- Adaptar según versiones específicas
- Configurar firewall
- Implementar autenticación
- Ajustar según infraestructura específica

### POSIBLES MEJORAS:

- Implementar Kerberos
- Configurar alta disponibilidad
- Implementar políticas de seguridad
- Optimizar configuraciones de red

¿Quieres que profundice en alguno de estos pasos o tienes alguna duda sobre la implementación?

### Puedes explicarme con más detalle las posibles mejoras?

Profundizaré en las posibles mejoras para un cluster de Hadoop y Spark:

#### 1. ALTA DISPONIBILIDAD (HA)

##### A. NameNode HA

Configuración:

- Dos NameNodes (activo/pasivo)
- Apache ZooKeeper para gestión
- Failover automático
- Shared Edits Storage

Configuración básica:

```
```xml
<configuration>
  <property>
    <name>dfs.ha.automatic-failover.enabled</name>
    <value>true</value>
  </property>
  <property>
    <name>dfs.nameservices</name>
    <value>mycluster</value>
  </property>
  <property>
    <name>dfs.ha.namenodes.mycluster</name>
    <value>nn1,nn2</value>
  </property>
</configuration>
````
```

- B. ResourceManager HA
  - Dos instancias de ResourceManager
  - Failover mediante ZooKeeper
  - Gestión de trabajos sin interrupciones

## 2. SEGURIDAD

### A. Kerberos

Implementación:

- Autenticación robusta
- Control de acceso
- Cifrado de comunicaciones

Pasos:

```
```bash
# Instalar paquetes
sudo apt-get install krb5-user libpam-krb5 libkadm55

# Configurar KDC (Key Distribution Center)
sudo apt-get install krb5-kdc krb5-admin-server

# Crear reino Kerberos
sudo krb5_newrealm

# Configurar principals para Hadoop
kadmin.local
> addprinc hdfs/master@MIDOMINIO
> addprinc yarn/master@MIDOMINIO
```

```

### B. Configuraciones de Seguridad

```
```xml
<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value>
</property>
<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
</property>
```

```

## 3. CIFRADO DE DATOS

### A. Cifrado en Reposo

- Usar dm-crypt/LUKS
- Cifrado de sistema de archivos
- Protección de datos sensibles

### B. Cifrado en Tránsito

- TLS para comunicaciones
- Certificados X.509
- Configuración de SSL/TLS

Ejemplo configuración:

```
```xml
<property>
```

```
<name>hadoop.ssl.enabled</name>
<value>true</value>
</property>
``
```

#### 4. MONITORIZACIÓN AVANZADA

##### A. Herramientas:

- Prometheus
- Grafana
- ELK Stack
- Apache Atlas

Ejemplo configuración Prometheus:

```
```yaml
scrape_configs:
  - job_name: 'hadoop'
    static_configs:
      - targets:
          - 'master:9870' # NameNode
          - 'slave1:9864' # DataNode
```
``
```

#### 5. OPTIMIZACIÓN DE RENDIMIENTO

##### A. Configuraciones Avanzadas

```
```xml
<!-- Aumentar tamaño de bloque --&gt;
&lt;property&gt;
  &lt;name&gt;dfs.blocksize&lt;/name&gt;
  &lt;value&gt;256M&lt;/value&gt;
&lt;/property&gt;

<!-- Optimizar memoria YARN --&gt;
&lt;property&gt;
  &lt;name&gt;yarn.nodemanager.resource.memory-mb&lt;/name&gt;
  &lt;value&gt;40960&lt;/value&gt;
&lt;/property&gt;
```
``</pre>
```

##### B. Tuning de Red

- Configurar interfaces 10GB
- Habilitar RDMA
- Optimizar TCP

#### 6. BALANCEO Y GESTIÓN DE CARGAS

##### A. YARN Queue Management

- Configurar colas de prioridad
- Límites de recursos por usuario/grupo

```
```xml
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>default,production,development</value>
</property>
```
``
```

## B. Dynamic Resource Allocation (Spark)

```

```
spark.dynamicAllocation.enabled true  
spark.shuffle.service.enabled true  
```
```

## 7. RECUPERACIÓN DE DESASTRES

### A. Backup y Replicación

- Configurar snapshots de HDFS
- Replicación entre clusters
- Backup incrementales

Script básico:

```
```bash  
#!/bin/bash  
BACKUP_DIR="/backup/hadoop"  
HDFS_PATH="/user/data"  
  
hdfs dfs -copyToLocal $HDFS_PATH $BACKUP_DIR/$(date +%Y%m%d)  
```
```

## 8. GESTIÓN DE CONTENEDORES

### A. Kubernetes + Hadoop

- Orquestación de contenedores
- Despliegue dinámico
- Escalado automático

### B. Docker Swarm

- Cluster de Docker
- Despliegue de servicios Hadoop

## 9. COMPLIANCE Y AUDITORÍA

### A. Apache Ranger

- Políticas de seguridad centralizadas
- Control de acceso fino
- Auditoría de accesos

### B. Apache Sentry

- Autorización a nivel de servidor
- Integración con Hive/Impala

## 10. MACHINE LEARNING OPS

### A. Kubeflow

- Pipelines de ML
- Integración con Spark
- Gestión de experimentos

### CONSIDERACIONES FINALES:

- No implementar todo a la vez
- Evaluar necesidades específicas
- Realizar pruebas incrementales
- Documentar cada cambio

Pasos Recomendados:

1. Seguridad básica
2. Monitorización
3. Alta disponibilidad
4. Optimización
5. Recuperación

Herramientas Complementarias:

- Apache Ambari (Gestión)
- Cloudera Manager
- Apache NiFi (Flujos de datos)

¿Quieres que profundice en alguna de estas mejoras o tienes un caso de uso específico?

**Me interesa hacer un ejemplo de Pipeline ETL en el que introduzca por ejemplo Kafka, Spark, hadoop, Zookeeper... qué me recomiendas que use? Tendría que ser para análisis en streaming**

Voy a diseñar un ejemplo completo de Pipeline ETL de streaming con las tecnologías que mencionas:

#### ARQUITECTURA DE PIPELINE ETL EN STREAMING

Componentes:

1. Fuente de Datos: Kafka
2. Procesamiento: Spark Streaming
3. Almacenamiento: Hadoop HDFS
4. Orquestación: Zookeeper
5. Gestión: Opcional (Airflow)

DIAGRAMA DE FLUJO:

```

Fuente Datos -> Kafka -> Zookeeper -> Spark Streaming -> HDFS -> Data Warehouse/Análisis

```

#### EJEMPLO PRÁCTICO: Análisis de Logs en Tiempo Real

Caso de Uso:

- Capturar logs de aplicación web
- Procesar en tiempo real
- Almacenar para análisis posterior
- Generar métricas en streaming

#### CÓDIGO DE EJEMPLO (Python/PySpark):

```
```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

# Configuración Spark Streaming
spark = SparkSession.builder \
    .appName("WebLogProcessing") \
    .getOrCreate()

# Esquema de Logs
```

```

log_schema = StructType([
    StructField("timestamp", TimestampType(), True),
    StructField("ip", StringType(), True),
    StructField("url", StringType(), True),
    StructField("status", IntegerType(), True)
])

# Conexión Kafka
df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "web_logs") \
    .load()

# Transformación
parsed_logs = df.select(
    from_json(col("value").cast("string"), log_schema).alias("log")
).select("log.*")

# Métricas en Tiempo Real
metrics = parsed_logs.groupBy(
    window(col("timestamp"), "1 minute"),
    col("status")
).count()

# Escritura a HDFS
query = metrics \
    .writeStream \
    .outputMode("complete") \
    .format("parquet") \
    .option("path", "hdfs://master:9000/logs/metrics") \
    .option("checkpointLocation", "/checkpoint") \
    .start()

query.awaitTermination()
```

```

## CONFIGURACIONES ADICIONALES

### 1. Kafka (docker-compose.yml):

```

```yaml
version: '3'
services:
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2181:2181"

  kafka:
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: localhost
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
```

```

```

2. Productor de Logs (Python):
```python
from kafka import KafkaProducer
import json
from datetime import datetime

producer = KafkaProducer(
    bootstrap_servers=['localhost:9092'],
    value_serializer=lambda x: json.dumps(x).encode('utf-8')
)

def send_log(ip, url, status):
    log = {
        "timestamp": datetime.now().isoformat(),
        "ip": ip,
        "url": url,
        "status": status
    }
    producer.send('web_logs', log)

# Ejemplo de uso
send_log("192.168.1.100", "/home", 200)
```

```

## INFRAESTRUCTURA RECOMENDADA

Servicios:

- 1 Nodo Zookeeper
- 2-3 Brokers Kafka
- 1 Nodo Spark Master
- 2-3 Nodos Spark Worker
- Cluster Hadoop con HDFS

## MEJORES PRÁCTICAS:

### 1. Tolerancia a Fallos

- Replicación en Kafka
- Checkpointing en Spark
- Replicación HDFS

### 2. Monitorización

- Prometheus
- Grafana
- ELK Stack

### 3. Seguridad

- Autenticación Kerberos
- Cifrado SSL
- Control de accesos

## OPCIONES DE ESCALAMIENTO:

Horizontales:

- Aumentar particiones Kafka
- Añadir nodos Spark
- Expandir cluster Hadoop

Verticales:

- Aumentar RAM
- Mejores CPUs
- SSDs de alto rendimiento

#### HERRAMIENTAS COMPLEMENTARIAS:

- Apache NiFi (Orquestación)
- Airflow (Scheduling)
- Superset (Visualización)
- Druid (Análisis en tiempo real)

#### CONSIDERACIONES DE COSTOS:

Cloud vs On-Premise:

- AWS: EMR, MSK
- Azure: HDInsight
- GCP: Dataproc

#### PASOS DE IMPLEMENTACIÓN:

1. Configurar Zookeeper
2. Instalar Kafka
3. Configurar Spark Streaming
4. Desplegar en Hadoop
5. Implementar monitorización
6. Añadir capa de seguridad

#### PRÓXIMOS PASOS RECOMENDADOS:

1. Definir caso de uso específico
2. Diseñar esquema de datos
3. Implementar infraestructura básica
4. Desarrollar pipeline incremental
5. Añadir complejidad gradualmente

¿Te gustaría que profundice en alguno de estos aspectos o tienes un caso de uso más concreto en mente?

**Quiero realizar los pasos de implementación de manera rápida y creo que docker podría ser una excelente forma de hacerlo. Me puedes ayudar a desplegarlo? Puedes escribir ahora todo en Inglés?**

I'll provide a comprehensive Docker-based approach to deploy a streaming ETL pipeline with Kafka, Spark, Hadoop, and Zookeeper.

#### DOCKER COMPOSITION STRATEGY

1. Create `docker-compose.yml`:

```
```yaml
version: '3.8'
services:
```

```

# Zookeeper Service
zookeeper:
  image: wurstmeister/zookeeper
  ports:
    - "2181:2181"
  networks:
    - data_network

# Kafka Service
kafka:
  image: wurstmeister/kafka
  ports:
    - "9092:9092"
  environment:
    KAFKA_ADVERTISED_HOST_NAME: kafka
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
  depends_on:
    - zookeeper
  networks:
    - data_network

# Hadoop NameNode
namenode:
  image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
  environment:
    - CLUSTER_NAME=hadoop_cluster
  ports:
    - "9870:9870"
  volumes:
    - hadoop_namenode:/hadoop/dfs/name
  networks:
    - data_network

# Hadoop DataNode
datanode:
  image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
  depends_on:
    - namenode
  volumes:
    - hadoop_datanode:/hadoop/dfs/data
  environment:
    - SERVICE_PRECONDITION=namenode:9870
  ports:
    - "9864:9864"
  networks:
    - data_network

# Spark Master
spark-master:
  image: bde2020/spark-master:3.1.1-hadoop3.2
  ports:
    - "8080:8080"
    - "7077:7077"
  environment:
    - INIT_DAEMON_STEP=setup_spark
  networks:
    - data_network

```

```

# Spark Worker
spark-worker:
  image: bde2020/spark-worker:3.1.1-hadoop3.2
  depends_on:
    - spark-master
  environment:
    - SPARK_MASTER=spark://spark-master:7077
  ports:
    - "8081:8081"
  networks:
    - data_network

# Optional: Airflow for Orchestration
airflow:
  image: apache/airflow:2.3.0
  environment:
    - AIRFLOW__CORE__EXECUTOR=LocalExecutor
    -
    AIRFLOW__DATABASE__SQLALCHEMY_CONN=postgresql+psycopg2://airflow:airflow@postgres/air
  flow
  ports:
    - "8082:8080"
  networks:
    - data_network

networks:
  data_network:
    driver: bridge

volumes:
  hadoop_namenode:
  hadoop_datanode:
  ...

```

## 2. Spark Streaming Python Script:

```

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

def create_spark_session():
    return SparkSession.builder \
        .appName("StreamingETLPipeline") \
        .config("spark.hadoop.fs.defaultFS", "hdfs://namenode:9000") \
        .getOrCreate()

def process_stream():
    spark = create_spark_session()

    # Log Schema Definition
    log_schema = StructType([
        StructField("timestamp", TimestampType(), True),
        StructField("ip", StringType(), True),
        StructField("url", StringType(), True),
        StructField("status", IntegerType(), True)
    ])

```

```

    ])

# Kafka Stream
df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "web_logs") \
    .load()

# Data Transformation
parsed_logs = df.select(
    from_json(col("value").cast("string"), log_schema).alias("log")
).select("log.*")

# Real-time Metrics
metrics = parsed_logs.groupBy(
    window(col("timestamp"), "1 minute"),
    col("status")
).count()

# Write to HDFS
query = metrics \
    .writeStream \
    .outputMode("complete") \
    .format("parquet") \
    .option("path", "hdfs://namenode:9000/logs/metrics") \
    .option("checkpointLocation", "/checkpoint") \
    .start()

query.awaitTermination()

if __name__ == "__main__":
    process_stream()
...

```

### 3. Kafka Log Producer:

```

```python
from kafka import KafkaProducer
import json
from datetime import datetime
import time
import random

def create_producer():
    return KafkaProducer(
        bootstrap_servers=['kafka:9092'],
        value_serializer=lambda x: json.dumps(x).encode('utf-8')
    )

def generate_log():
    status_codes = [200, 404, 500, 302]
    return {
        "timestamp": datetime.now().isoformat(),
        "ip": f"192.168.1.{random.randint(1, 255)}",
        "url": random.choice(["/home", "/product", "/login", "/api"]),
    }

```

```

        "status": random.choice(status_codes)
    }

def produce_logs():
    producer = create_producer()

    while True:
        log = generate_log()
        producer.send('web_logs', log)
        time.sleep(1) # 1 log per second

if __name__ == "__main__":
    produce_logs()
...

```

#### DEPLOYMENT STEPS:

##### 1. Install Docker and Docker Compose

```

```bash
sudo apt update
sudo apt install docker.io docker-compose
sudo usermod -aG docker $USER
```

```

##### 2. Create Project Structure

```

```bash
project_root/
|
└── docker-compose.yml
└── spark_streaming.py
└── kafka_producer.py
└── README.md
```

```

##### 3. Start Services

```

```bash
docker-compose up -d
```

```

##### 4. Verify Services

```

```bash
docker-compose ps
```

```

##### 5. Run Spark Streaming Job

```

```bash
docker-compose exec spark-master spark-submit \
    --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.1 \
    spark_streaming.py
```

```

##### 6. Run Kafka Producer

```

```bash
docker-compose exec kafka python kafka_producer.py
```

```

#### MONITORING AND MANAGEMENT:

- Hadoop UI: <http://localhost:9870>
- Spark UI: <http://localhost:8080>
- Airflow UI: <http://localhost:8082>

#### RECOMMENDATIONS:

##### 1. Security

- Add SSL/TLS
- Implement Kerberos
- Use Docker Secrets

##### 2. Performance Tuning

- Adjust Kafka partitions
- Optimize Spark configurations
- Use SSD volumes

##### 3. Monitoring

- Integrate Prometheus
- Add Grafana dashboards
- Implement log aggregation

##### 4. Scalability

- Add more Kafka brokers
- Increase Spark workers
- Use cloud-native solutions

#### NEXT STEPS:

- Define specific use case
- Design data schema
- Implement advanced transformations
- Add machine learning components

Would you like me to elaborate on any specific aspect of this implementation?

# lsBash LINUX CMD

jueves, 12 de diciembre de 2024 10:11

Explicación:

## 1. Prompt

Nombre de usuario@máquina en la que estoy: carpeta actual

Es personalizable

## 2. Working with directories:

- a. Where am I?: **pwd**
- b. Change directory: **cd** absolute or relative (use tab to help your self autofill)
- c. Make directory: **mkdir**
- d. List files in a directory: **ls**, for more info. Ls -lah (list, all (included hidden, human readable size))
- e. Clean the screen: **clear**

## 3. Working with files:

- a. Create files:
  - i. **nano** (ribbon down to save...)
  - ii. Vi (more complicated, but very powerfull. No user friendly. No down ribbon). To exit: Press Esc key, :w (safe), :q (exit)
- b. Copy files:
  - i. copy one file to a path: cp file1.txt relative//absolute\_path
  - ii. Copy several files: cp \*.txt path/
- c. Move files:
  - i. same as copy but with mv. Will move the file.
- d. information in files:
  - i. show content of file:
    - 1) cat
    - 2) head -3 file1.txt
    - 3) tail -3 file2.txt
    - 4) number lines, words, letters: wc file1.txt
      - a) wc -l file1.txt (lines)
      - b) wc -w file1.txt (words)
    - 5) Pipes: cat file1.txt | head -3 | tail -1 (this extracts line 3)
      - a) save to another file: head -3 file1.txt > out.txt
        - i) if you use it again, you over write
        - ii) if you use it with >> (doble), you append
      - b) Pipes in other direction:
        - i) tail -4 < out.txt
  - e. permissions:  

    - i. first: d: directory
    - ii. 3 groups:
      - 1) first: owner: read, write, execute
      - 2) second: owner's group
      - 3) others: not owner and not from owners group
    - iii. meanings of letters:
      - 1) read, write, execute
    - iv. erase/give permissions:
      - 1) chmod -r file1.txt (user can't read file)

- 2) chmod u+r file1.txt (give user permission to read)  
 3) chmod u+x file1.txt (give user permission to execute--> important to execute scripts without having to call sudo (superuser))  
 4) chmod o-w file1.txt  
 f. processes: (memory usage, etc...)  
     i. see: ps  
     ii. more (ps -v)  
     iii. even more (ps -e v)  
     iv. sleep  
     v. kill

| Command | Description                 | Example                                    | Additional Parameters                                                                                     |
|---------|-----------------------------|--------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| ls      | List files and directories  | ls shows basic directory contents          | ls -l (detailed list), ls -la (includes hidden files), ls -lh (human-readable file sizes)                 |
| cd      | Change directory            | cd /home/user moves to specified directory | cd .. (move up one directory), cd ~ (go to home directory), cd - (return to previous directory)           |
| pwd     | Print working directory     | pwd displays current directory path        | pwd -P (resolves symbolic links), pwd -L (shows logical path)                                             |
| mkdir   | Create directories          | mkdir new_folder creates a directory       | mkdir -p parent/child (creates parent directories if needed), mkdir -v (verbose output)                   |
| rm      | Remove files or directories | rm file.txt deletes a file                 | rm -r (recursive deletion), rm -f (force removal), rm -rf (force remove directories)                      |
| cp      | Copy files or directories   | cp file1.txt file2.txt copies file         | cp -r (recursive copy), cp -i (interactive mode), cp -p (preserves file attributes)                       |
| mv      | Move or rename files        | mv oldname.txt newname.txt renames         | mv -i (interactive mode), mv -u (update only newer files), mv -v (verbose output)                         |
| touch   | Create or update files      | touch newfile.txt creates file             | touch -a (only update access time), touch -m (only modify time), touch -d (set specific date)             |
| cat     | Display file contents       | cat file.txt shows file content            | cat -n (number lines), cat -b (number non-blank lines), cat file1.txt file2.txt (concatenate files)       |
| grep    | Search text patterns        | grep "pattern" file.txt                    | grep -i (case-insensitive), grep -r (recursive search), grep -n (show line numbers)                       |
| chmod   | Change file permissions     | chmod 755 script.sh                        | chmod u+x (add execute for user), chmod go-w (remove write for group/others), chmod -R (recursive change) |
| sudo    | Execute with admin rights   | sudo apt update                            | sudo -i (interactive root shell), sudo -u username (run as specific user)                                 |
| man     | Display command manual      | man ls shows command details               | man -k keyword (search commands), man section number (specific manual section)                            |
| find    | Search files                | find /home -name "*.txt"                   | find . -type f (only files), find . -type d (only directories), find -size +10M (files larger than 10MB)  |
| tar     | Archive files               | tar -cvf archive.tar files/                | tar -xvf (extract), tar -zvf (extract compressed), tar -czvf (create compressed archive)                  |
| ssh     | Connect to remote           | ssh user@hostname                          | ssh -p port (specify port), ssh -i keyfile (use identity file), ssh -v (verbose mode)                     |

crontab

ecrontab

# Scripting

jueves, 12 de diciembre de 2024 6:11

[Tutorial de inmersión al Terminal Linux y Bash Scripting](#)



<https://www.albertcoronado.com/2021/03/11/procesar-ficheros-de-datos-json-y-csv-en-bash-python-y-java-big-data/>

<https://www.albertcoronado.com/2021/03/02/comandos-bash-para-trabajar-con-grandes-ficheros-de-datos-bigdata-para-principiantes/>

# Docker

jueves, 12 de diciembre de 2024 7:10

Docker (engine)

[How to Install Docker on Ubuntu: A Step-By-Step Guide](#)



Docker Desktop Installation



On a server: engine and client. No need to ssh. Directly execute commands on Linux server  
[Docker Engine for Linux Servers Setup and Tips](#)



# Kafka

lunes, 16 de diciembre de 2024 11:58

Presentation: [https://www.canva.com/design/DAGZcwh8mEM/Yh2MKzwoPSHQnES95SVn2Q/edit?utm\\_content=DAGZcwh8mEM&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=sharebutton](https://www.canva.com/design/DAGZcwh8mEM/Yh2MKzwoPSHQnES95SVn2Q/edit?utm_content=DAGZcwh8mEM&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)

Overview  
Hardware needed  
Data abstraction  
Installation  
Practical use cases  
Real world use cases - hands on exercises  
Integration with other tools  
Troubleshooting

First ChatGPT prompt

## 1. Overview

Presenter: Jean Alvarez KLK

- Content:
  - What is Apache Kafka?
  - History and development by LinkedIn; now part of the Apache Software Foundation.
  - Core features: distributed event streaming, fault tolerance, scalability.
  - Use cases: message queues, real-time analytics, log aggregation, etc.
  - Kafka's architecture: Producers, Consumers, Brokers, Topics, Partitions.
  - Diagram: Kafka's ecosystem and data flow.

## 2. Hardware Needed

Presenter: Lukas

- Content:
  - System requirements for Kafka deployment:
    - Minimum and recommended CPU, memory, and disk space.
    - Network requirements (high throughput, low latency).
  - Disk considerations: SSDs vs HDDs, RAID configurations for durability.
  - Kafka broker capacity planning (messages/day, throughput per broker).
  - Zookeeper requirements (until version 3.3+).
  - Recommendations for cloud infrastructure (e.g., AWS, Azure, GCP).

## 3. Data Abstraction

Presenter: Alejandro

- Content:
  - Key concepts in Kafka's data model:
    - Topics:** Logical streams of messages.
    - Partitions:** Subdivision of topics for scalability.
    - Offset:** Identifier for messages within a partition.
    - Producers and Consumers:** Writing to and reading from topics.
  - Kafka's **log-based storage** model and immutability.
  - Message serialization: JSON, Avro, Protocol Buffers.
  - Schema evolution and compatibility.

## 4. Installation

Presenter: Youssef

- Content:
  - Prerequisites: Java (JDK), Zookeeper (for older versions), Kafka binaries.
  - Step-by-step guide:
    - Downloading Kafka.
    - Setting up Zookeeper or KRaft mode.
    - Configuring Kafka broker properties (server.properties).
    - Starting Kafka and Zookeeper services.
    - Testing the setup with a producer/consumer CLI example.
  - Deployment options:
    - Local setup vs. distributed cluster.
    - Kafka on Kubernetes and cloud-based deployments.

## 5. Practical and Real-World Use Cases

Presenter: Vaughan

- Content:

- Practical Scenarios:

- Real-time log monitoring, event-driven microservices, and ETL pipelines using Kafka Connect.

- Real-world Applications:

- Examples from Netflix, Uber, and LinkedIn on personalization, event processing, and activity stream processing.

## 6. Integration with Other Tools

Presenter: Jorge

- Content:

- Kafka Connect ecosystem:

- Integrating with databases (PostgreSQL, MySQL).
    - Stream processing frameworks: Apache Flink, Spark Streaming.
    - Monitoring with Prometheus and Grafana.

- Kafka with REST APIs (Confluent REST Proxy).

- Overview of Confluent Kafka: Added features and tools.

- Using Kafka with cloud-native services (AWS MSK, Azure Event Hubs).

## 7. Troubleshooting

Presenter: Aman

- Content:

- Common issues:

- Consumer lags, message loss, Zookeeper timeouts.
    - Partition leader election issues.

- Debugging tools:

- Kafka logs, Kafka Manager, and command-line utilities.

- Checklist for diagnosing problems:

- Broker health, producer/consumer configs.
    - Zookeeper connectivity (or KRaft issues for newer Kafka versions).

- Preventive measures:

- Topic cleanup policies, retention periods.
    - Monitoring tools for proactive alerting.

### Additional Notes:

1. Presentation Style:

- Use clear and visually appealing slides (diagrams, charts, minimal text).
  - Include live demos (for practical examples and hands-on exercises).

2. Collaboration Tools:

- Use Google Slides or Microsoft PowerPoint for collaboration.
  - Assign tasks early and set deadlines for review.

3. Preparation:

- Rehearse collectively to ensure seamless transitions between sections.
  - Prepare for Q&A with extra notes or slides for commonly asked questions.

Let me know if you need help with specific content or slide design!

# Overview

lunes, 16 de diciembre de 2024 11:58

- Content:

## What is Apache Kafka?

Apache Kafka is an open-source, distributed event streaming platform designed to handle real-time data feeds. It is used for building real-time data pipelines and streaming applications. Kafka efficiently manages and processes high-throughput, low-latency streams of data in a fault-tolerant manner.

## History and development by LinkedIn; now part of the Apache Software Foundation.

Apache Kafka was initially developed by LinkedIn in 2010 to manage real-time data streams effectively. It became an open-source project in 2011 and was later donated to the Apache Software Foundation in 2012, becoming one of its most popular projects. Today, Kafka is widely adopted by organizations for its robust capabilities in event streaming and data processing.

## Core features: distributed event streaming, fault tolerance, scalability.

### Distributed Event Streaming

- Kafka distributes data across multiple servers, enabling efficient, scalable streaming of events in real time.

### Fault Tolerance

- Kafka replicates data across multiple brokers to ensure durability and availability, even in the event of server failures.

### Scalability

- Kafka's partitioned log model allows for horizontal scalability by distributing partitions across brokers.

### High Performance

- Kafka achieves high throughput for both publishing and subscribing, even with many concurrent producers and consumers.

### Durability

- Data is written to disk and replicated, ensuring it remains accessible over time.

## Use cases: message queues, real-time analytics, log aggregation, etc.

### Message Queues

- Kafka is often used as a highly reliable, distributed message broker.

### Real-Time Analytics

- Processes streaming data to gain insights or make predictions in real time.

### Log Aggregation

- Collects and centralizes logs from multiple systems for monitoring and debugging.

### Event Sourcing

- Captures all changes to an application state as a series of events.

### Stream Processing

- Enables transformations and analysis of data streams using frameworks like Kafka Streams or Apache Flink.

### IoT Applications

- Handles data from thousands of connected devices in real time.

## Kafka's architecture: Producers, Consumers, Brokers, Topics, Partitions.

Key Components:

### Producers

- Applications that send data to Kafka topics.

### Consumers

- Applications that read data from Kafka topics.

### Brokers

- Servers that store and serve Kafka data.

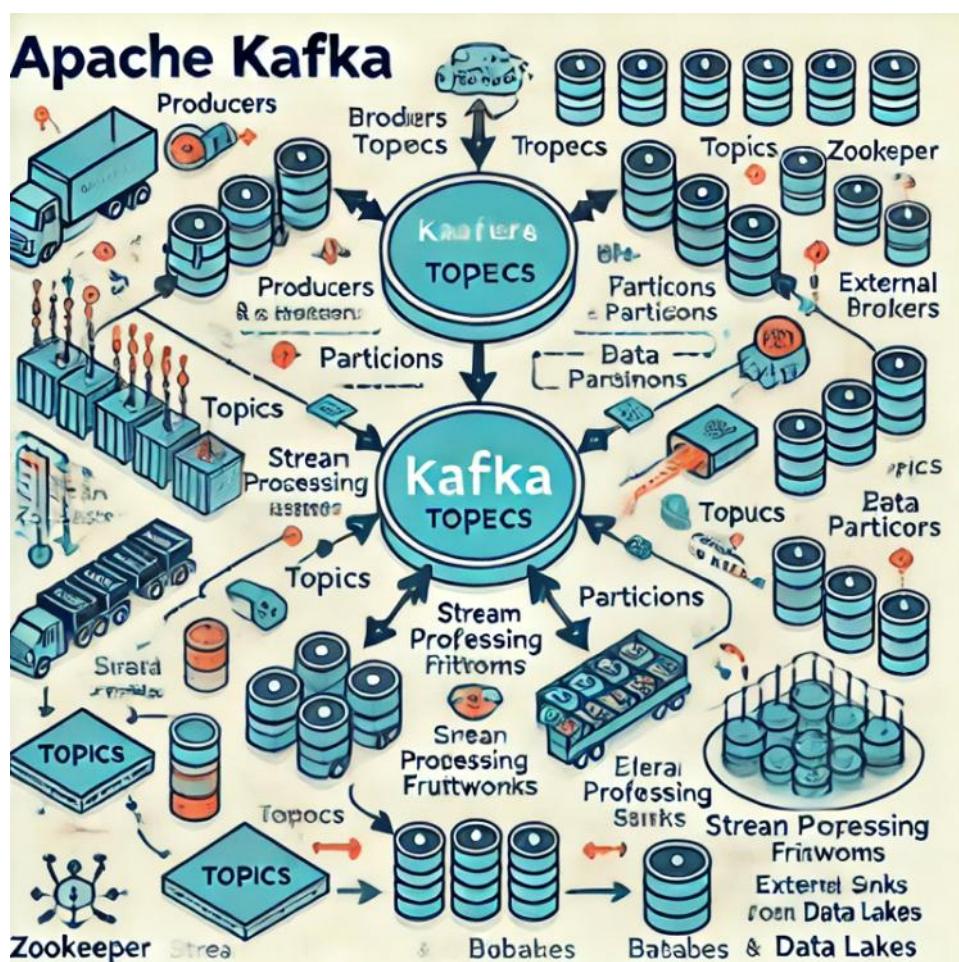
### Topics

- Categories or channels to which data is sent. Topics are split into partitions.

### Partitions

- Subdivisions of topics that enable parallelism and scalability. Each partition is an ordered, immutable sequence of records.

## Diagram: Kafka's ecosystem and data flow.



Apache Kafka was born out of a need for a **high-throughput, low-latency, distributed messaging system** capable of handling real-time data streams. It was developed by **LinkedIn** in 2010 and later open-sourced in 2011. Here are the key motivations behind Kafka's creation:

## 1. Real-Time Data Processing at Scale

- LinkedIn needed a system to process massive amounts of real-time event data generated by its platform (e.g., user interactions, log events, and system metrics).
- Existing systems (like traditional message queues) couldn't handle the scale or required too much overhead for large-scale, fault-tolerant real-time applications.

## 2. Decoupling Systems

- As LinkedIn grew, it needed a way to decouple producers (systems generating data) and consumers (systems processing data).
- Kafka was designed to act as a central hub for streaming data, making it easier to integrate and scale disparate systems.

## 3. Durability and Fault Tolerance

- A persistent, fault-tolerant system was necessary to ensure data was not lost even in the event of failures.
- Kafka achieves this with its distributed design, replication of data across brokers, and write-ahead logs.

## 4. Event Sourcing and Replay

- LinkedIn wanted the ability to **replay events** for debugging, analytics, or system recovery purposes.
- Kafka stores data in a **log-like structure** that allows consumers to read and re-read messages at their own pace, enabling such capabilities.

## 5. High Performance

- The need for high throughput and low latency to process millions of events per second was a critical factor.
- Kafka achieves this by leveraging efficient disk I/O operations, partitioning, and a streamlined binary protocol.

## 6. Open Source and Community Growth

- By open-sourcing Kafka, LinkedIn aimed to foster a larger community, improve the technology through contributions, and encourage adoption across industries.

# Data abstraction

Lunes, 16 de diciembre de 2024 12:05

Key concepts in Kafka data model:

Topics: Logical streams of messages

A topic in Kafka is a logical channel where messages are written by producers and from which messages are consumed by consumers. Think of it as a category where data is sent.

Partitions: Subdivision of topics for scalability

Think of the partitions as a storage room for the topics, partitions help Kafka to be able to handle massive amount of data by dividing the topics among the different partitions

Offset: Identifier for messages within a partition

Each message has an ID and is called offset. The whole point of this is to keep track of where you left off in a partition. Think of it as the page of a book that you are reading

Producers and consumers: writing to and reading from topics.

Producers are the apps or any kind of software that send information to Kafka, then send them to a topic.

Consumers are the clients or subscribers (as it is call in Kafka), they read the data from Kafka subscribing to topics and process the messages.

Kafka log-based storage model and immutability:

Once something is written in a partitions stay there until it gets deleted after a certain period (depending on how you configure it). This ensure that all of the consumers can process the data in the same order it was written.

Message serialization: JSON, Avro and Protocol Buffers.

Serialization is the act of convert the data so it can be send over the wire, you have to convert it into a format compatible with Kafka, such as JSON, Avro and Protocol Buffers (Protobuf)

Schema Evolution and Compatibility

You are able to make changes in your data structure, this is thanks to Schema evolution and compatibility, but you have to be aware of some rules that you have to keep so your structure doesn't break  
Some of the rules are

Never remove fields

Add fields with default values

Change field types carefully

There's 3 types of compatibility

Backward compatibility: New version can read old data

Forward compatibility: Old consumers can read new data

Full Compatibility: Both forward and backward compatibility are ensured.

## Data Abstraction in Kafka

### Presenter: Alejandro

Apache Kafka's data abstraction layer is foundational for understanding how data flows through its ecosystem. Let's explore the key components of Kafka's data model, its storage mechanics, and the role of serialization in ensuring seamless data processing.

### Key Concepts in Kafka's Data Model

#### 1. Topics

- Topics represent logical streams of messages in Kafka.
- They act as categories or feed names to which producers send messages and consumers subscribe.

#### 2. Partitions

- Topics are subdivided into partitions to enable horizontal scalability.
- Each partition is an ordered sequence of messages that Kafka distributes across brokers.

#### 3. Offset

- An offset uniquely identifies each message within a partition.
- It acts as a pointer, enabling consumers to track their position and resume processing seamlessly.

#### 4. Producers and Consumers

- **Producers** are responsible for writing data to Kafka topics.
- **Consumers** read data from topics, typically using offsets to manage their progress.

### Kafka's Log-Based Storage Model and Immutability

- Kafka stores messages in a log format within partitions.
- Messages are immutable once written, ensuring consistency and reliability.
- Logs are segmented and retained based on user-configurable policies, such as time-based or size-based limits.

### Message Serialization

Serialization is key to encoding messages for efficient transmission and storage. Kafka supports several serialization formats, including:

- **JSON**: Human-readable and widely used but less efficient in terms of size.
- **Avro**: Compact and schema-based, enabling smaller message sizes and efficient data parsing.
- **Protocol Buffers (Protobuf)**: Highly efficient and used in scenarios demanding high performance.

### Schema Evolution and Compatibility

- Kafka integrates schema registries to manage schema evolution seamlessly.
- Schemas define the structure of messages and ensure compatibility across producers and

consumers.

- Compatibility strategies include:
  - **Backward Compatibility:** Consumers using older schemas can process messages produced with newer schemas.
  - **Forward Compatibility:** Producers using older schemas can write messages that newer consumers understand.
  - **Full Compatibility:** Ensures both backward and forward compatibility, maintaining system integrity during updates.

# Installation

lunes, 16 de diciembre de 2024 12:17

## Step-by-Step Guide to Install Apache Kafka on Ubuntu

### Prerequisites

#### 1. \*\*Java (JDK):\*\*

- Ensure Java is installed and configured.

- Download: <https://www.oracle.com/java/technologies/javase-downloads.html> (Oracle JDK) or <https://openjdk.org/> (OpenJDK).

#### 2. \*\*Zookeeper (for Kafka < 2.8.0):\*\*

- Required for older Kafka versions.

- Download: <https://zookeeper.apache.org/>.

#### 3. \*\*Kafka Binaries:\*\*

- Download the appropriate version from: <https://kafka.apache.org/downloads>.

### 1. Downloading Kafka

- Extract the downloaded Kafka archive:

```
tar -xzf kafka_{version}.tgz
```

```
cd kafka_{version}
```

### 2. Setting Up Zookeeper or KRaft Mode

- \*\*For Kafka < 2.8.0:\*\*

- Start Zookeeper:

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

- \*\*For Kafka >= 2.8.0:\*\*

- Use KRaft mode (no Zookeeper required):

```
bin/kafka-storage.sh format -t <UUID> -c config/kraft/server.properties
```

### 3. Configuring Kafka Broker Properties

- Edit `config/server.properties`:

- Set `broker.id=0`.

- Update `log.dirs=/tmp/kafka-logs`.

- Configure listeners:

```
listeners=PLAINTEXT://:9092
```

```
advertised.listeners=PLAINTEXT://<your-ip>:9092
```

Note: To find your IP address, use `ifconfig` or `ip addr show` in the terminal.

### 4. Starting Kafka and Zookeeper

- Start Zookeeper (if required):

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

- Start Kafka Broker:

```
bin/kafka-server-start.sh config/server.properties
```

### 5. Testing the Setup

- Create a topic:

```
bin/kafka-topics.sh --create --topic test-topic --bootstrap-server localhost:9092
```

- Start a producer:

```
bin/kafka-console-producer.sh --topic test-topic --bootstrap-server localhost:9092
```

Enter messages in the terminal.

- Start a consumer:

```
bin/kafka-console-consumer.sh --topic test-topic --from-beginning --bootstrap-server localhost:9092
```

Verify the producer's messages appear in the consumer terminal.

## 6. Deployment Options

- **Local Setup vs. Distributed Cluster:**

- Local setups are simpler for testing and development.

- Distributed clusters improve scalability and reliability for production.

- **Kafka on Kubernetes:**

- Use frameworks like Strimzi (<https://strimzi.io/>) or Confluent Operator (<https://www.confluent.io/confluent-operator/>).

- **Cloud-Based Deployments:**

- Managed Kafka services: AWS MSK, Azure Event Hubs, or Confluent Cloud.

# Hardware needed

lunes, 16 de diciembre de 2024 12:06

## For Kafka Deployment:

### 1. Kafka Cluster Hardware Components

Apache Kafka is typically deployed as a **cluster of nodes (servers)**.

#### a. Broker Nodes

- Kafka operates with **broker nodes**, which are responsible for receiving, storing, and managing the data streams. These brokers form the heart of a Kafka cluster.
  - **Number of Brokers:** The minimum setup should have **3 brokers** to ensure fault tolerance. In larger setups, you might have **5 or more brokers**.
  - **Hardware Requirements:**
    - **CPU:** Kafka brokers handle a lot of I/O operations, so **multiple CPU cores** (e.g., 4–8 cores) are recommended to handle incoming and outgoing data streams efficiently.
    - **Memory (RAM):** Kafka brokers need sufficient RAM for efficient data processing. A typical configuration would include **16–32 GB of RAM**.
    - **Storage:** Start with **500 GB** of disk space for a small Kafka broker setup. For high-throughput setups, **1–3 TB** of disk space per broker are recommended.
    - **Network:** A **fast network connection** (e.g., **10Gbps**) is necessary to handle high-throughput data streaming.

#### b. Zookeeper Nodes

- Apache Kafka relies on **Zookeeper** to manage its cluster metadata, handle leader election for partitions, and coordinate the brokers. Zookeeper needs its own set of nodes.
  - **Number of Zookeeper Nodes:** Typically, **3 nodes** (or more) for high availability.
  - **Hardware Requirements:**
    - **CPU:** Zookeeper is lightweight, so **2–4 CPU cores** should be enough for typical use cases.
    - **Memory (RAM):** **8–16 GB** of RAM
    - **Storage:** Zookeeper doesn't require high storage, but it still needs fast disks for metadata storage.

#### c. Producer and Consumer Nodes

- **Producers** send data to Kafka brokers, and **Consumers** retrieve data from Kafka. These are typically client applications or services.
  - **Hardware Requirements:** The hardware required for producers and consumers depends on the nature of the data and the amount of processing needed. Generally:
    - **CPU:** Moderate CPU resources, depending on data processing requirements.
    - **Memory (RAM):** 8–16 GB should be enough for most consumer/producer setups.
    - **Network:** Fast network connections to interact with the Kafka brokers.

#### **CPU:**

- **Minimum:** A basic server with at least a **2–4 core CPU** should work for small Kafka clusters.
- **Recommended:** For larger clusters, especially with higher throughput, **8–16 cores** are recommended, because Kafka is highly CPU-intensive

#### **Memory (RAM):**

- **Minimum:** Kafka can run with **4–8 GB RAM** for small deployments or testing environments.
- **Recommended:** For production use, Kafka brokers usually require **16–64 GB** of RAM, especially if handling a high volume of messages.

#### **Disk Space:**

- **Minimum:** Start with **500 GB** of disk space for a small Kafka broker setup.
- **Recommended:** For high-throughput setups, **1-3 TB** of disk space per broker are recommended.

#### ***Network Requirements:***

- Kafka performs best with **high-throughput, low-latency network**. Ideally, the network should support **10 Gbps** speeds.
- Kafka is designed to handle bursts of data efficiently, so **latency** should be as low as possible (ideally under **10ms**) between brokers and producers/consumers.

# Integration with Other Tools

lunes, 16 de diciembre de 2024 12:06

- **BIG DATA AND STREAM PROCESSING**

Kafka allows real-time data streaming and processing in big data ecosystems. Its ability to handle high-throughput, fault-tolerant messaging makes it a perfect fit for integrating with various tools.

- Apache Hadoop
- Apache Spark
- Apache Flink
- Apache Storm
- Kafka Streams
- Apache Samza

- **MACHINE LEARNING**

It enables continuous data flow, model inference, and model retraining pipelines allowing to unlock capabilities for predictive analytics, anomaly detection, and decision-making in real-time.

- TensorFlow
- PyTorch
- Seldon

- **CLOUD SERVICES**

Integrating Kafka with cloud data services and stream processing tools enables advanced real-

time data pipelines and event-driven architectures.

- Confluent Cloud
- AWS
- GCP
- Azure

- **DATA WAREHOUSE AND BI TOOLS**

Kafka serves as the backbone for data ingestion, streaming data into data warehouses, while BI tools provide an interactive interface for visualizing and analyzing that data.

- Snowflake
- Tableau
- Power BI

# Troubleshooting

18 December 2024 10:37

## Common Issues:

- **Consumer Lags:** Delay in processing messages by consumers.
- **Message Loss:** Messages not reaching consumers due to misconfigurations or failures.
- **Zookeeper Timeouts:** Connectivity issues between Kafka and Zookeeper leading to unstable cluster states.
- **Partition Leader Election Issues:** Problems in assigning a leader for partitions, often caused by broker failures or unbalanced clusters.

## Debugging Tools:

- **Kafka Logs:** Check broker logs, consumer logs, and producer logs for error messages and stack traces.
- **Kafka Manager:** A UI tool to monitor and manage Kafka clusters, useful for visualizing partitions and consumer groups.
- **Command-line Utilities:**
  - kafka-consumer-groups.sh: Inspect consumer group offsets and lags.
  - kafka-topics.sh: Manage and describe topics.
  - kafka-broker-api-versions.sh: Check compatibility and protocol versions.

## Checklist for Diagnosing Problems:

1. **Broker Health:**
  - Verify broker uptime and resource utilization (CPU, memory, disk).
  - Ensure brokers are evenly handling partitions.
2. **Producer/Consumer Configurations:**
  - Check producer acknowledgments (acks) and retry settings.
  - Validate consumer auto.offset.reset and group ID configurations.
3. **Zookeeper Connectivity:**
  - Confirm stable Zookeeper quorum and connectivity.
  - For newer Kafka versions using KRaft, verify the controller quorum health.

## Preventive Measures:

- **Topic Cleanup Policies:**
  - Configure log compaction or retention periods to manage storage effectively.
  - Use partitions and replication factors that align with the system's fault tolerance requirements.
- **Monitoring Tools:**
  - Use tools like Prometheus, Grafana, or Confluent Control Center for proactive alerting.
  - Set up alerts for key metrics like consumer lag, broker disk usage, and partition unavailability.

# SPARK

lunes, 16 de diciembre de 2024 13:31

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/index.html>

```
docker run -p 8888:8888 -p 4040:4040 -p 4041:4041 -e JUPYTER_ENABLE_LAB=yes -v  
"/c/Users/marti/OneDrive - EAE Business School/Big Data Processing/Shared with  
Students/BIGDATA-Examples/SPARK:/home/jovyan/work" jupyter/pyspark-notebook
```

```
docker run -p 8888:8888 -p 4040:4040 -p 4041:4041 -e JUPYTER_ENABLE_LAB=yes -v  
"/Users/alejandrovidalsuarez/Library/CloudStorage/OneDrive-EAEBusinessSchool/Shared with  
Students/Students Sand Boxes/Alejandro Suárez/SPARK:/home/jovyan/work" jupyter/pyspark-  
notebook
```

José Luis Martínez Arribas. <http://127.0.0.1:8888/lab?token=940c5be52679bbdcda33a6bcc050ab9c3aaac9c9cfb354e7>

# ETL- FB,KAFKA,PYTHON,POSTGRES

martes, 14 de enero de 2025 21:28

## Creating a Data Pipeline with KPIs: Step-by-Step Guide Using Fluent Bit, Kafka, Python, and Postgres

This article explains how to build a **Data Pipeline** for generating KPIs using technologies like Fluent Bit, Kafka, Python, and Postgres.

A **data pipeline** is a logical construction representing a process divided into phases. In this case, the process begins with ingesting a real Nginx log file, publishing it to a Kafka queue, consuming it with a Python script that stores KPIs in a Postgres database. Finally, we'll use an application called **CloudBeaver** to query the data, completing the cycle:

Data source → Processing → Storage → Usage.

### Infrastructure

The required infrastructure for this example includes:

1. **Zookeeper + Kafka** for message streaming.
2. **Postgres** as the database system.
3. **CloudBeaver**, a web-based database client.

We'll use the following docker-compose.yml file to set up the environment:

```
docker-compose.yml
version: '3.8'

services:
  zookeeper:
    container_name: zookeeper
    image: bitnami/zookeeper:latest
    ports:
      - "2181:2181"
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes

  kafka:
    container_name: kafka
    image: bitnami/kafka:latest
    ports:
      - "9092:9092"
      - "29092:29092"
    environment:
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,EXTERNALPLAINTEXT://:29092
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka:9092,EXTERNALPLAINTEXT://localhost:29092
      - KAFKA_LISTENER_SECURITY_PROTOCOL_MAP=PLAINTEXT:PLAINTEXT,EXTERNALPLAINTEXT:PLAINTEXT
```

```

    - KAFKA_INTER_BROKER_LISTENER_NAME=EXTERNALPLAINTEXT
    - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
    - ALLOW_PLAINTEXT_LISTENER=yes
depends_on:
  - zookeeper

```

```

postgres:
  container_name: postgres
  image: postgres
  environment:
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: postgres
    POSTGRES_DB: postgresdb
    PGDATA: /data/postgres
  ports:
    - "5432:5432"
  volumes:
    - /home/joseluis/data/postgres-data:/data/postgres
  restart: unless-stopped

```

```

cloudbeaver:
  container_name: cloudbeaver
  image: dbeaver/cloudbeaver:latest
  ports:
    - "8080:8978"

```

Run the services using the command:

```
docker-compose up
```

## Initializing the Database

Open CloudBeaver to manage the database (create password, login and run the following scripts)

Create a schema and table in Postgres to store KPIs:

Run 1st Script:

```
CREATE SCHEMA test;
```

Run 2nd Script:

```

CREATE TABLE test.kpis (
  kpi_key VARCHAR(100) NOT NULL,
  kpi_value BIGINT NOT NULL,
  vcount INTEGER NOT NULL,
  PRIMARY KEY ( kpi_key, kpi_value )
);

```

**Data structure:**

- **kpi\_key:** KPI name (e.g., "Requests per minute").
- **kpi\_value:** KPI value (e.g., "minute 15, hour 10, day 12, August 2023").
- **vcount:** Request count (e.g., 24).

## Sample Log Data

Here are some example log lines you can use for testing:

```
163.116.184.104 - - [23/May/2023:14:21:08 +0000] "POST /data.php?f1=1
&url=https://www.albertcoronado.com/2016/07/04/configurando-un-
balanceador-para-liferay-con-nginxseguridad-tlsdocker/ HTTP/1.1" 200 9258
"https://www.albertcoronado.com/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-tlsdocker/" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0
Safari/537.36" "83.45.66.225"
```

```
163.116.184.104 - - [23/May/2023:14:21:08 +0000] "GET /img/lostsys.png
HTTP/1.1" 200 5635
"[https://www.albertcoronado.com/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-tlsdocker/
(https://www.albertcoronado.com/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-tlsdocker/)" "Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0
Safari/537.36" "83.45.66.225"
```

```
163.116.184.104 - - [23/May/2023:14:21:08 +0000] "GET /wp-
content/uploads/2016/07/nginx-liferay-balancer.png HTTP/1.1" 200 74619
"[https://www.albertcoronado.com/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-tlsdocker/
(https://www.albertcoronado.com/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-tlsdocker/)" "Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0
Safari/537.36" "83.45.66.225"
```

```
163.116.184.104 - - [23/May/2023:14:21:11 +0000] "GET /robots.txt
HTTP/1.1" 200 92 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36"
"83.45.66.225"
```

```
107.22.135.11 - - [23/May/2023:14:21:12 +0000] "GET
/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-
tlsdocker HTTP/1.1" 301 408 "-" "Mozilla/5.0 (compatible; proximic;
+[https://www.comscore.com/Web-Crawler](https://www.comscore.com/Web-Crawler))" "-"
```

```
107.22.135.11 - - [23/May/2023:14:21:12 +0000] "GET
/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-
tlsdocker/ HTTP/1.1" 200 61431 "-" "Mozilla/5.0 (compatible; proximic;
+[https://www.comscore.com/Web-Crawler](https://www.comscore.com/Web-Crawler))" "-"
```

```
94.237.45.203 - - [23/May/2023:14:21:12 +0000] "GET
/2016/07/04/configurando-un-balanceador-para-liferay-con-nginxseguridad-
tlsdocker HTTP/1.1" 301 408 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36"
"-"
```

## Processing Logs with Fluent Bit

**Fluent Bit** reads, processes, and forwards data from sources like log files or databases. Here, it will read an `nginx.log` file and send data to a Kafka queue.

### 0. Install Fluent Bit (Linux):

```
curl https://raw.githubusercontent.com/fluent/fluent-bit/master/install.sh
| sh
```

test it is correctly installed

```
/opt/fluent-bit/bin/fluent-bit --h
```

1. Create file to Configure de Parser:

parser.conf:

```
[PARSER]
  Name    nginx
  Format  regex
  Regex ^(?<remote>[^ ]*) (?<host>[^ ]*) (?<user>[^ ]*) \[(?<time>[^ ]*)\] "(?<method>\S+)(?: +(?<path>[^"]*?)(?: +\$*)?)?" (?<code>[^ ]*)(?<size>[^ ]*)(?: "(?<referer>[^"]*)" "(?<agent>[^"]*)")"
  Time_Key time
  Time_Format %d/%b/%Y:%H:%M:%S %z
```

2. Create file to Configure the log execution:

fluent-bit.conf

```
[INPUT]
  name tail
  read_from_head true
  exit_on_eof true
  path /opt/posts/data/logs/nginx.log
  parser nginx
```

```
[OUTPUT]
  Name kafka
  Match *
  Brokers 127.0.0.1:29092
  Topics test
```

3. Run Fluent Bit:

```
/opt/fluent-bit/bin/fluent-bit -c ./fluent-bit.conf -R ./parser.conf
```

## Python Script for Processing Kafka Data

Here's the Python script to consume Kafka messages and store KPIs in Postgres:

Requirements.txt

```
kafka-python === 2.0.2
psycopg2 === 2.9.6
```

Before installing:

```
sudo apt update
sudo apt install postgresql libpq-dev
```

Create a folder named python in your project

```
pip install -r requirements.txt
```

Script de python

```
import datetime
import json
```

```

import logging
from kafka import KafkaConsumer
import psycopg2

year_part = 1
month_part = 2
day_part = 3
hour_part = 4
minute_part = 5
def build_kpi_value( timestamp, parts ):
    date = str( datetime.datetime.fromtimestamp( timestamp ) ).split( " " )
    dateparts = date[ 0 ].split("-") + date[ 1 ].split(":")
    return "".join( dateparts[0:parts] )
def store_kpi( kpi_key, kpi_value, vcount ):
    sql = """
        INSERT INTO test.kpis(kpi_key, kpi_value, vcount) VALUES ( %s, %s, %s )
        ON CONFLICT ( kpi_key, kpi_value )
        DO UPDATE SET vcount = kpis.vcount + EXCLUDED.vcount;
    """
    cur.execute( sql, (kpi_key, kpi_value, vcount) )
    conn.commit()
conn = psycopg2.connect(
    database="postgresdb",
    host="127.0.0.1",
    user="postgres",
    password="postgres",
    port="5432"
)
cur = conn.cursor()
consumer = KafkaConsumer(
    group_id='my-group',
    bootstrap_servers=['127.0.0.1:29092'],
    value_deserializer=lambda x: json.loads(x.decode("utf-8"))
)
consumer.subscribe( topics = [ 'test' ] )
for msg in consumer: logging.info( "{0}".format( msg ) )
    store_kpi(
        "REQUEST_X_MINUTE",
        build_kpi_value( msg.value["@timestamp"], minute_part ),
        1
    )
    store_kpi(
        "REQUEST_X_HOUR",
        build_kpi_value( msg.value["@timestamp"], hour_part ),
        1
    )
    store_kpi(
        "REQUEST_X_DAY",
        build_kpi_value( msg.value["@timestamp"], day_part ),
        1
    )

```

## Querying Data in CloudBeaver

Complete the pipeline by querying the data via SQL using CloudBeaver at <http://127.0.0.1:8080>. Alternatively, consider integrating dashboards with tools like Grafana, Superset, or Jupyter.

This tutorial demonstrates a complete cycle for a data pipeline. Feedback, suggestions, or improvements (e.g., adding dashboards or other KPI types) are always welcome!

# fluent bit in windows

martes, 14 de enero de 2025 6:51

Install docker  
docker-compose.yml

docker-compose up

postgres ui to add password  
cbadmin  
Cbadmin1

login

create schema  
create table

install fluent-bit:

- Linux (script in the web):
  - o curl <https://raw.githubusercontent.com/fluent/fluent-bit/master/install.sh> | sh
- Windows(exe):  
<https://docs.fluentbit.io/manual/installation/windows#installing-from-exe-installer>

```
PS> .\bin\fluent-bit.exe -i dummy -o stdout
Fluent Bit v2.0.x
* Copyright (C) 2019-2020 The Fluent Bit Authors
* Copyright (C) 2015-2018 Treasure Data
* Fluent Bit is a CNCF sub-project under the umbrella of Fluentd
* https://fluentbit.io
```

```
[2019/06/28 10:13:04] [ info] [storage] initializing...
[2019/06/28 10:13:04] [ info] [storage] in-memory
[2019/06/28 10:13:04] [ info] [storage] normal synchronization mode, checksum
disabled, max_chunks_up=128
[2019/06/28 10:13:04] [ info] [engine] started (pid=10324)
[2019/06/28 10:13:04] [ info] [sp] stream processor started
[0] dummy.0: [1561684385.443823800, {"message":>"dummy"}]
[1] dummy.0: [1561684386.428399000, {"message":>"dummy"}]
[2] dummy.0: [1561684387.443641900, {"message":>"dummy"}]
[3] dummy.0: [1561684388.441405800, {"message":>"dummy"}]
```

# Best Practices

miércoles, 15 de enero de 2025 10:37

VM:

- OS: Ubuntu is well suited for Big Data Purposes
  - Resources: Ram (half of the free suggested)
  - Threads (CPU): top 1/2 of the free suggested
  - Disk Space allocation: 2x of the needed and never. Leave 30Gb free of disk space in your host computer:
    - o 30 Gb for a short test, 50 Gb docker deployments

Installations:

- Update & upgrade
- Give superuser permissions for not having to run always sudo (if you know what you are doing)
- Python:
  - o virtual environment for each project (allows not to mess your configuration)
- File Management:
  - o folder for each project:
    - Folder Structure:
      - Name of the project:
        - ◆ Data
        - ◆ Logs
        - ◆ Scripts (where you are going to place the coding)
        - ◆ Configuration files
        - ◆ .env (virtual environment variables --> to not commit to git)
        - ◆ Virtual Environment Folder (where python and libs are located)

# AIRFLOW

lunes, 20 de enero de 2025 10:06

[Tutorial de Apache Airflow: Desde puesta en marcha, crear DAGs, cliente CLI y API](#)



## CHAPTER 4

lunes, 13 de enero de 2025 10:27

# Types of Big Data processing to model business logic

# Business Logic Case - kms/truck/year

miércoles, 22 de enero de 2025 9:03

## **Task Description:**

You are data analysts working for a transportation company, assisting area managers with reporting tasks.

The fleet manager reaches out to you, explaining that they need to annually submit information on the **total kilometers traveled by each truck to a government agency**.

The company has vehicles registered under two separate entities:  
**Company A** and **Company B**.

Unfortunately, the telemetry data for the vehicles was lost during a database migration. You come up with the idea of retrieving this information from the daily CSV reports that are emailed.

You manage to extract these files from the emails and save them in the directory:

ETL-Python/data

Teams of 4.

## **Phase I:**

1. Analyze the downloaded data and determine which files contain the information needed to meet the fleet manager's requirements.
2. Create a single CSV file combining all relevant files (using Python) and save it in your sandbox at the following location:  
ETL-Python/data/Output/typefilename\_combined.csv
  - i. Each team will work in 2 pairs:  
2 people uses python and 2 PowerQuery (Excel).
3. Analyze the combined content and provide a summary of the information.

## **Phase II:**

Design the Data Architecture to ensure the company collects the information needed to their use

# Documentation Coding

sábado, 25 de enero de 2025 19:30

Devdocs.io

Free (open source)

lunes, 13 de enero de 2025 10:27

Postgres.new

Crear bases de datos con IA

# Big Data Processing

2410-22\_2MaBDBA\_FT-EN-02A

October 2024- Máster in Big Data & Business Analytics

Teacher: José Luis Martínez Arribas

De:

 Planeta Formación y Universidades

eae.es

# COURSE PRESENTATION

## 1. Goals:

- A. Data management: basic concepts and fundamentals.
- B. Introduction to massive data processing: infrastructures, types, development and applications.
- C. Application deployment. Development of scalable applications.
- D. Types of Big Data processing to model business logic.
- E. Models, architectures, tools, and high-level languages for massive data processing.

# METHODOLOGY

---

## Evaluate

1. Course Attendance
2. Individual and Group Contribution, in class and in the Collaborative Notebook
3. Submission and Presentation of Collaborative and Individual Activities in BB
4. Written Test on Learned Concepts

# INTRODUCTION



**José Luis Martínez  
Arribas**

Industrial Engineer with master's degrees in Logistics & Supply Chain and Business Analytics.

Since 2003, Co-Founder of Ensaco Eficiencia Energética, a renewable energy engineering and installation company specializing in smart control systems, business consultant, and trainer in BA, BI, and Big Data.

He has over 15 years of experience in the logistics and transportation sector, as well as in renewable energy installations and smart building control systems.

¿Why this Master?

YOUR GOALS

## Info conjunta

- [EMAILS AND GOALS](#)



## Notebook Colaborativo



[ONENOTE COMPARTIDO](#)

# Big Data Processing

2410-22\_2MaBDBA\_FT-EN-02A

October 2024- Máster in Big Data & Business Analytics

Teacher: José Luis Martínez Arribas

De:

 Planeta Formación y Universidades

eae.es

# COURSE PRESENTATION

## 1. Goals:

- A. Data management: basic concepts and fundamentals.
- B. Introduction to massive data processing: infrastructures, types, development and applications.
- C. Application deployment. Development of scalable applications.
- D. Types of Big Data processing to model business logic.
- E. Models, architectures, tools, and high-level languages for massive data processing.

# METHODOLOGY

---

## Evaluate

1. Course Attendance
2. Individual and Group Contribution, in class and in the Collaborative Notebook
3. Submission and Presentation of Collaborative and Individual Activities in BB
4. Written Test on Learned Concepts

# INTRODUCTION



**José Luis Martínez  
Arribas**

Industrial Engineer with master's degrees in Logistics & Supply Chain and Business Analytics.

Since 2003, Co-Founder of Ensaco Eficiencia Energética, a renewable energy engineering and installation company specializing in smart control systems, business consultant, and trainer in BA, BI, and Big Data.

He has over 15 years of experience in the logistics and transportation sector, as well as in renewable energy installations and smart building control systems.

¿Why this Master?

YOUR GOALS

## Info conjunta

- [EMAILS AND GOALS](#)



## Notebook Colaborativo



[ONENOTE COMPARTIDO](#)

# Big Data Processing

2410-22\_2MaBDBA\_FT-EN-02A

October 2024- Máster in Big Data & Business Analytics

Teacher: José Luis Martínez Arribas

De:

 Planeta Formación y Universidades

eae.es

## Content

- A. Data management: basic concepts and fundamentals.
- B. Introduction to massive data processing: infrastructures, types, development and applications.
- C. Application deployment. Development of scalable applications.
- D. Types of Big Data processing to model business logic.
- E. Models, architectures, tools, and high-level languages for massive data processing.

# 01. Data management: basic concepts and fundamentals.

---

# What does DATA MANAGEMENT mean?

---

## Data Management

Is the process of collecting, storing, organizing, and maintaining data to ensure it's accessible, accurate, and ready for analysis.

It means understanding how to handle data throughout its lifecycle, from raw data collection to processing and storage, all the way to preparing it for decision-making insights.

# Key Concepts

## 1. Data Collection:

Gathering data from various sources, like customer databases, sales records, or social media, ensuring it's relevant and comprehensive for the business problem at hand.

## 2. Data Storage:

Using systems (like databases or data warehouses) to store data securely and systematically. This includes cloud storage solutions that make large-scale data management feasible and scalable.

## 3. Data Cleaning and Preparation:

Ensuring data quality by removing duplicates, fixing errors, and handling missing values so that analyses are accurate and reliable.

## Key Concepts

### **4. Data Governance and Security:**

Establishing policies for data access, privacy, and compliance to protect sensitive information and meet regulatory requirements.

### **5. Data Integration:**

Combining data from multiple sources, like CRM systems or marketing platforms, to get a holistic view for analysis.

### **6. Data Access and Analytics:**

Making data accessible to the right people at the right time, often through dashboards or analytics tools, to support data-driven decision-making.

Understanding these elements helps to effectively use data as a strategic asset, making it easier to derive insights and make informed business decisions.

## 1. Data Collection

**What It Is:** This is the foundational step where you gather data relevant to your business questions or objectives. Data can come from internal sources (like sales records, CRM databases, financial systems) or external sources (like market research, social media, or economic data).

### Key Steps:

- **Identify Data Sources:** Determine where the data comes from, including transaction systems, customer feedback forms, IoT devices, or third-party APIs. In business analytics, data sources should align with your business needs.
- **Define Data Types:** Decide if you need structured data (like tables in a database) or unstructured data (like social media posts). Structured data is easier to analyze, while unstructured data often requires pre-processing but can reveal insights like customer sentiment.
- **Select Collection Methods:** Common methods include automated data pipelines (for transactional or real-time data), surveys (for customer preferences), and web scraping (for collecting publicly available data). Choose methods based on accuracy, reliability, and ease of integration.
- **Ensure Ethical and Legal Compliance:** Be mindful of data privacy laws like GDPR in Europe or CCPA in California. Always obtain data responsibly and, where applicable, anonymize it to protect individual privacy.

**Why It Matters:** Good data collection practices ensure you have reliable data that represents the real-world phenomenon you're studying. It's the backbone of any analysis and ultimately impacts the accuracy and quality of your insights.

## 2. Data Storage

**What It Is:** After collecting data, you need a safe, organized space to store it. Storage solutions vary based on data size, type, and access requirements.

### Key Steps:

- **Choose Storage Solutions:** Options include databases (like MySQL or PostgreSQL for relational data), data warehouses (like Snowflake for large, structured data), and data lakes (like Amazon S3 for raw or semi-structured data).
- **Consider Cloud Storage:** Cloud solutions (e.g., AWS, Google Cloud, Azure) offer scalable, cost-effective storage and make data accessible from anywhere. They're also convenient for big data projects that require storage flexibility.
- **Organize Data Structure:** Organize your data logically. Data should be easy to locate and access, so apply structures (e.g., database schemas, table names) that facilitate analysis.
- **Ensure Data Backup and Security:** Backup mechanisms prevent data loss, and security measures (like encryption and access controls) protect against unauthorized access.

**Why It Matters:** Proper storage ensures data is accessible, secure, and ready for analysis. It allows for efficient processing and retrieval, especially as the volume of data grows.

### 3. Data Cleaning and Preparation

**What It Is:** Also known as “data wrangling” this step ensures that data is in good shape for analysis by fixing errors, standardizing formats, and filling in missing information.

#### Key Steps:

- **Remove Duplicates:** Identify and eliminate any duplicate records that could distort analysis.
- **Fix Data Quality Issues:** Correct inconsistencies (e.g., formatting differences) and errors (like typos or outliers).
- **Handle Missing Data:** Decide how to address gaps. You can remove incomplete rows, fill in missing values (with averages, for example), or use imputation techniques.
- **Transform Data for Analysis:** Sometimes, data must be transformed into a suitable format (e.g., converting dates into standard formats or splitting text fields) for further processing.

**Why It Matters:** Clean, high-quality data leads to accurate, reliable analysis, reducing the risk of misleading conclusions.

## 4. Data Governance and Security

**What It Is:** This step involves setting policies and standards to manage data access, privacy, and security, ensuring that only authorized users can access specific data.

### Key Steps:

- **Define Access Controls:** Use role-based access to restrict data based on user roles, keeping sensitive information secure.
- **Set Privacy and Compliance Standards:** Data must comply with laws and regulations (e.g., GDPR, HIPAA) and should respect user privacy.
- **Create a Data Usage Policy:** Outline how data should be used, shared, and stored within the organization. Policies help prevent misuse and ensure data integrity.
- **Implement Security Protocols:** Use encryption, secure passwords, and regular audits to protect data against breaches.

**Why It Matters:** Governance and security ensure data is protected, usable, and compliant with legal standards, safeguarding both the business and its stakeholders.

## 5. Data Integration

**What It Is:** Data integration involves combining data from multiple sources into a cohesive, centralized format, enabling a comprehensive view of the business.

### Key Steps:

- **Establish Common Data Definitions:** Ensure that data fields are consistent across sources. For example, if “customer ID” appears in multiple databases, it should follow the same format.
- **Use ETL Tools:** Extract, Transform, Load (ETL) tools like Talend or Informatica facilitate data extraction, cleaning, and loading into a central repository.
- **Ensure Data Synchronization:** Data should be updated regularly across systems so that information remains accurate and up-to-date.
- **Resolve Data Conflicts:** Handle discrepancies (e.g., different names for the same customer across databases) to ensure that all data aligns.

**Why It Matters:** Integrated data gives a complete picture of business operations, supporting better analytics and decision-making by combining insights from multiple data sources.

## 6. Data Access and Analytics

**What It Is:** The final step is to make data accessible to users (often through dashboards or reports) so they can analyze and derive actionable insights.

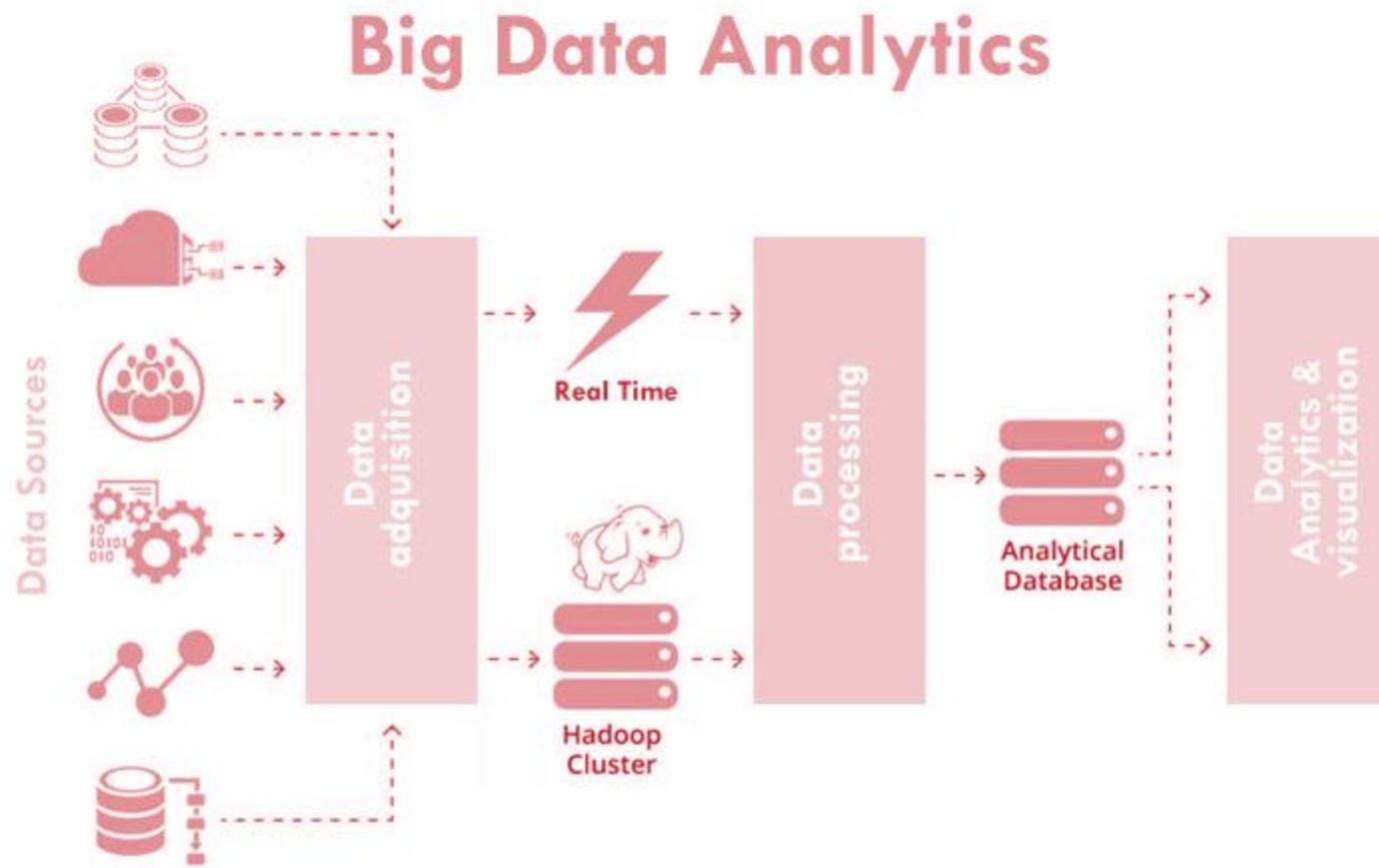
### Key Steps:

- **Implement Business Intelligence (BI) Tools:** Tools like Power BI, Tableau, or Looker visualize data for business users, making insights accessible and digestible.
- **Ensure Role-Based Data Access:** Only authorized users should access specific data, maintaining privacy and security.
- **Enable Self-Service Analytics:** Providing tools and resources for business users to perform their analyses can enhance decision-making across departments.
- **Measure Key Metrics and KPIs:** Define relevant metrics, such as customer retention rates or sales growth, to monitor performance and drive decisions.

**Why It Matters:** Proper storage ensures data is accessible, secure, and ready for analysis. It allows for efficient processing and retrieval, especially as the volume of data grows.

# Data management: basic concepts and fundamentals. DATA LIFECYCLE

## Data Life Cycle



## Example

### GEOTRACKING COMPANY:

**1. Devices in trucks send data to cloud:**

**2. App To Show Almost Real Time:**

- Truck/Drivers distance (Kms)
- Time in each status (driving, working, waiting & available)
- Documents

**3. ETL Pipeline to aggregate all collected valuable information**

**4. Data Storage: Data Warehouse for analytical purposes**

- Reporting services

## Deep review of the tool

- INFORMATION NEEDED
  - Distance
  - Speed
  - GPS
  - Brake
  - Fuel Consumption...
- INFORMATION AVAILABLE
  - Sensors:
    - Truck:
      - [CANBUS](#)
    - Trailer:
      - [CANBUS](#)
      - THERMOGRAPH



# Data management: basic concepts and fundamentals. DATA STORAGE

## Data Storage

1. Introduction to Data Storage
2. Relational Databases: SQL
3. No Relational Databases: NoSQL
4. Data Warehouses
5. Data Lakes
6. Case Study
7. Wrap-Up and Q&A

## Data Storage

### 1. Introduction to Data Storage

“Data is key to good decisions, but great data unlocks **powerful insights** that propel impactful actions”

Goal: efficient data storage for business insights

#### **Key Points:**

- There are different storage systems: databases, data warehouses, data lakes.
- Choosing the right storage for efficient data access and analysis.

**Discussion:** personal experiences with data storage (e.g., spreadsheets, cloud storage).

- **Bases de datos relacionales**

- Se utilizan en: aplicaciones tradicionales, DataWarehouse, ERP, CRM y e-commerce.
- Productos

## Data Storage

### 1. Relational Databases: SQL

Almacenan datos cuyas relaciones y esquema están predefinidos, diseñadas para admitir transacciones ACID y conservar la integridad referencial, así como la coherencia de los datos.

Database structure: tables, rows, columns, primary and foreign keys.

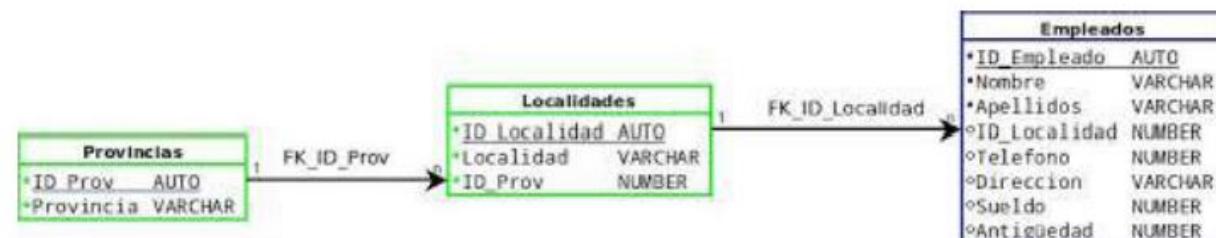
SQL basics: SELECT, INSERT, UPDATE, DELETE.

#### Products:

- MySQL, PostgreSQL, MariaDB, Oracle, SQL
- Amazon Aurora, Amazon RDS, Amazon

#### Assistance:

AI that helps to create a database:



**Discussion:** Simple SQL exercise on a sample database (e.g., employee or transportation data).

# Incapacidad de manejo de los datos del Método Tradicional

## Relational Database Management Systems (RDBMS)

- Terabytes y Petabytes de datos → no puede con ellos
  - Se necesitan cada vez Máquinas más potentes (más procesadores y memoria), y hacen poco viable su implementación
- 80% de los datos recopilados son semi-estructurados o no estructurados, con lo cual no puede analizarlos
- No puede lidiar con la velocidad de entrada de datos

## Data Mining vs Big Data

4 | 1 Introduction to the World of Big Data

Table 1.1 Differences in the attributes of big data and RDBMS.

| ATTRIBUTES    | RDBMS                  | BIG DATA                         |
|---------------|------------------------|----------------------------------|
| Data volume   | gigabytes to terabytes | petabytes to zettabytes          |
| Organization  | centralized            | distributed                      |
| Data type     | structured             | unstructured and semi-structured |
| Hardware type | high-end model         | commodity hardware               |
| Updates       | read/write many times  | write once, read many times      |
| Schema        | static                 | dynamic                          |

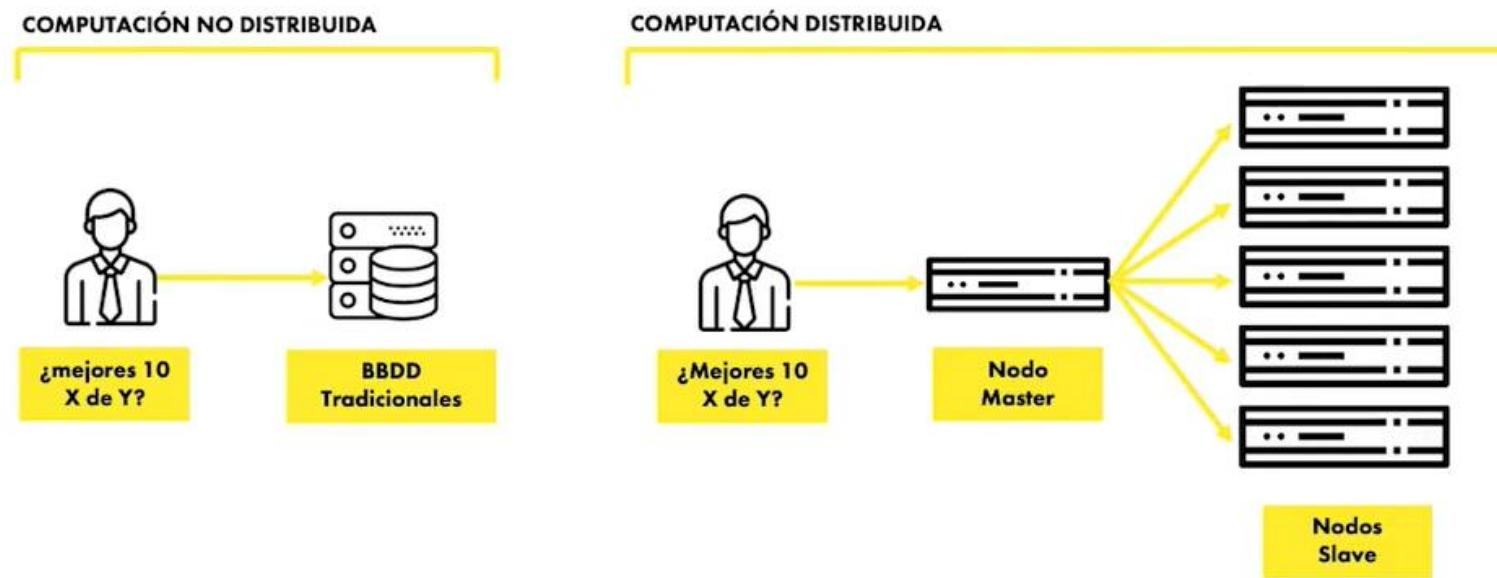
Table 1.2 Data Mining vs. Big Data.

| S. No. | Data mining                                                                                   | Big data                                                                                                                   |
|--------|-----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| 1)     | Data mining is the process of discovering the underlying knowledge from the data sets.        | Big data refers to massive volume of data characterized by volume, velocity, and variety.                                  |
| 2)     | Structured data retrieved from spread sheets, relational databases, etc.                      | Structured, unstructured, or semi-structured data retrieved from non-relational databases, such as NoSQL.                  |
| 3)     | Data mining is capable of processing large data sets, but the data processing costs are high. | Big data tools and technologies are capable of storing and processing large volumes of data at a comparatively lower cost. |
| 4)     | Data mining can process only data sets that range from gigabytes to terabytes.                | Big data technology is capable of storing and processing data that range from petabytes to zettabytes.                     |

# Distributed Storage

**Storage and processing of big volumes of data with speed and low cost.**

## DISTRIBUTED COMPUTING



## ¿Qué es BIG DATA?

### DISTRIBUTED STORAGE



# DATA STORAGE SOLUTIONS

## Most commonly used Big Data Storage Solutions

- Hadoop
- Elasticsearch
- Mongo db
- Hbase
- Cassandra
- Neo4J

Data management: basic concepts and fundamentals.  
**DATA CLEANING AND PREPARATION**

---

# Data Cleaning and Preparation

## Overview:

Often called "data wrangling," this process focuses on getting your data ready for analysis by correcting errors, harmonizing formats, and dealing with missing values.

## Key Processes:

- 1. Eliminate Duplicates:** Detect and remove repeated entries that could skew analytical results.
- 2. Resolve Data Quality Issues:** Address inconsistencies such as mismatched formats or errors, including typos and extreme outliers.
- 3. Address Missing Values:** Determine how to handle gaps in your data—options include removing incomplete entries, substituting missing values (e.g., with averages), or applying advanced imputation techniques.
- 4. Prepare Data for Analysis:** Modify and format data as needed, such as standardizing date formats, splitting combined text fields, or reorganizing datasets for easier analysis.

## Importance:

Accurate, well-prepared data ensures dependable insights and minimizes the risk of drawing incorrect conclusions.

# Data Cleaning and Preparation

**Beyond Basics:** Once the foundational cleaning is done, advanced techniques can further optimize your data for analysis and enhance the quality of your insights.

## Key Advanced Steps:

### 1. Feature Engineering:

1. Create new variables or modify existing ones to uncover additional insights.
2. Example: Deriving age from a date of birth or calculating a profitability ratio.

### 2. Normalization and Scaling:

1. Adjust numerical data to a common scale without distorting relationships.
2. Example: Scaling income data to fall between 0 and 1 for machine learning models.

### 3. Outlier Treatment:

1. Use statistical methods to identify and handle outliers that could bias results.
2. Approaches: Winsorization, clipping, or applying robust statistical techniques.

### 4. Data Enrichment:

1. Integrate additional data sources to provide more context.
2. Example: Augmenting sales data with weather information for trend analysis.

### 5. Automating the Process:

1. Leverage tools or scripts (e.g., Python, R) to automate repetitive cleaning and preparation tasks, improving efficiency and consistency.

**Why It's Critical:** Advanced preparation ensures your data is not just clean but also tailored to the analytical techniques you plan to use, enabling deeper insights and more effective decision-making.

## Data Cleaning and Preparation

### STEPS:

1. WHAT DO WE WANT? WHY WE ARE DOING THIS ANALYSIS?
2. WHAT DATA DO WE HAVE?

- UNDERSTANDING DATA
  - A. STRUCTURED AND SEMISTRUCTURED DATA
  - B. UNSTRUCTURED DATA

A: FIRST STEPS ARE GOING TO BE PERFORMED WITH AN ETL TOOL THAT WE ALL HAVE IN OUR COMPUTERS BUT NEVER NEW WE HAD IT.

We are going to dive in a tool that everybody has in its computer: POWERQUERY FOR EXCEL.

What we are going to learn is also valid for a business intelligence tool like POWER BI.

# Data Cleaning and Preparation

## STEPS:

1. What do we want? Why we are doing this analysis?
2. What data do we have?
  - Understanding data
    - A. Structured and semistructured data
    - B. Unstructured data
3. Beyond the basics: outlier treatment, new variables, enrich data
4. Automate: scripts r/Python
5. Connecting those scripts with Workflows or cronjobs/scheduled tasks

Data management: basic concepts and fundamentals.  
**DATA GOVERNANCE AND SECURITY**

---

## 4. DATA GOVERNANCE AND SECURITY

**Establishing policies for data access, privacy, and compliance** ensures sensitive information is protected and regulatory requirements are consistently met. This foundation supports trustworthy, ethical, and efficient use of data across the organization.

- **ACCESS CONTROL:** Defines clear permissions and roles to ensure that only authorized individuals can access specific datasets, minimizing the risk of unauthorized use or breaches.
- **PRIVACY:** Implements robust measures to safeguard personal and sensitive information, adhering to frameworks such as GDPR, HIPAA, or CCPA to build trust with stakeholders.
- **COMPLIANCE:** Aligns data management practices with legal and industry regulations, ensuring that data usage meets all standards for security, auditability, and accountability.
- **TRANSPARENCY:** Creates clear documentation and audit trails for how data is collected, stored, shared, and processed, promoting organizational integrity and readiness for regulatory audits.
- **RISK MANAGEMENT:** Proactively identifies vulnerabilities and establishes protocols to prevent, detect, and respond to potential data breaches or misuse.

Through strong governance and security practices, organizations can not only protect their data assets but also foster a culture of responsibility, mitigate risks, and enhance the reliability of their data-driven initiatives.

# Data management: basic concepts and fundamentals. DATA INTEGRATION

## 5. DATA INTEGRATION

**Combining data from multiple sources**, such as CRM systems, marketing platforms, or financial databases, creates a unified and holistic view of organizational information. This enables seamless analysis, improved collaboration, and more strategic decision-making.

- **CONNECTIVITY:** Ensures smooth and reliable access to various data sources, facilitating real-time or scheduled synchronization to keep information up-to-date across systems.
- **TRANSFORMATION:** Standardizes, cleanses, and enriches data during integration, ensuring consistency, accuracy, and usability for downstream analytics and reporting.
- **VISIBILITY:** Provides stakeholders with a comprehensive, 360-degree view of operations, customers, and performance by breaking down silos between disparate data systems.
- **EFFICIENCY:** Automates data workflows and reduces manual effort, accelerating the availability of integrated data for timely insights and action.

By enabling the seamless merging of data, integration supports scalable analytics, drives operational excellence, and aligns cross-functional objectives, all while ensuring compliance and governance of data assets.

# Data management: basic concepts and fundamentals. DATA ACCESS AND ANALYTICS

## 6. DATA ACCESS AND ANALYTICS

**Making data accessible to the right people at the right time** ensures that stakeholders can act on accurate, timely information, often through dashboards, reports, or analytics tools, fostering a culture of data-driven decision-making across the organization.

- **NOTIFICATIONS:** Serve as proactive alerts, ensuring that users are immediately informed of significant changes, anomalies, or opportunities within the data, enabling swift action and reducing response times.
- **REPORTS:** Provide structured, in-depth insights with formulas, graphs, and data visualizations, summarizing historical trends and key performance metrics to inform strategy and operations.
- **DASHBOARD:** Interactive dashboards consolidate and visualize data in real-time, making complex information more accessible and actionable. By bringing businesses closer to their data, dashboards empower users to monitor performance, identify patterns, and make informed decisions with agility.

This approach to data access and analytics aligns with broader data management goals, including ensuring data integrity, compliance, and scalability to support evolving business needs.

# 01. Data management: basic concepts and fundamentals.

---

# What does DATA MANAGEMENT mean?

---

## Data Management

Is the process of collecting, storing, organizing, and maintaining data to ensure it's accessible, accurate, and ready for analysis.

It means understanding how to handle data throughout its lifecycle, from raw data collection to processing and storage, all the way to preparing it for decision-making insights.

# Key Concepts

## 1. Data Collection:

Gathering data from various sources, like customer databases, sales records, or social media, ensuring it's relevant and comprehensive for the business problem at hand.

## 2. Data Storage:

Using systems (like databases or data warehouses) to store data securely and systematically. This includes cloud storage solutions that make large-scale data management feasible and scalable.

## 3. Data Cleaning and Preparation:

Ensuring data quality by removing duplicates, fixing errors, and handling missing values so that analyses are accurate and reliable.

## Key Concepts

### **4. Data Governance and Security:**

Establishing policies for data access, privacy, and compliance to protect sensitive information and meet regulatory requirements.

### **5. Data Integration:**

Combining data from multiple sources, like CRM systems or marketing platforms, to get a holistic view for analysis.

### **6. Data Access and Analytics:**

Making data accessible to the right people at the right time, often through dashboards or analytics tools, to support data-driven decision-making.

Understanding these elements helps to effectively use data as a strategic asset, making it easier to derive insights and make informed business decisions.

## 1. Data Collection

**What It Is:** This is the foundational step where you gather data relevant to your business questions or objectives. Data can come from internal sources (like sales records, CRM databases, financial systems) or external sources (like market research, social media, or economic data).

### Key Steps:

- **Identify Data Sources:** Determine where the data comes from, including transaction systems, customer feedback forms, IoT devices, or third-party APIs. In business analytics, data sources should align with your business needs.
- **Define Data Types:** Decide if you need structured data (like tables in a database) or unstructured data (like social media posts). Structured data is easier to analyze, while unstructured data often requires pre-processing but can reveal insights like customer sentiment.
- **Select Collection Methods:** Common methods include automated data pipelines (for transactional or real-time data), surveys (for customer preferences), and web scraping (for collecting publicly available data). Choose methods based on accuracy, reliability, and ease of integration.
- **Ensure Ethical and Legal Compliance:** Be mindful of data privacy laws like GDPR in Europe or CCPA in California. Always obtain data responsibly and, where applicable, anonymize it to protect individual privacy.

**Why It Matters:** Good data collection practices ensure you have reliable data that represents the real-world phenomenon you're studying. It's the backbone of any analysis and ultimately impacts the accuracy and quality of your insights.

## 2. Data Storage

**What It Is:** After collecting data, you need a safe, organized space to store it. Storage solutions vary based on data size, type, and access requirements.

### Key Steps:

- **Choose Storage Solutions:** Options include databases (like MySQL or PostgreSQL for relational data), data warehouses (like Snowflake for large, structured data), and data lakes (like Amazon S3 for raw or semi-structured data).
- **Consider Cloud Storage:** Cloud solutions (e.g., AWS, Google Cloud, Azure) offer scalable, cost-effective storage and make data accessible from anywhere. They're also convenient for big data projects that require storage flexibility.
- **Organize Data Structure:** Organize your data logically. Data should be easy to locate and access, so apply structures (e.g., database schemas, table names) that facilitate analysis.
- **Ensure Data Backup and Security:** Backup mechanisms prevent data loss, and security measures (like encryption and access controls) protect against unauthorized access.

**Why It Matters:** Proper storage ensures data is accessible, secure, and ready for analysis. It allows for efficient processing and retrieval, especially as the volume of data grows.

### 3. Data Cleaning and Preparation

**What It Is:** Also known as “data wrangling” this step ensures that data is in good shape for analysis by fixing errors, standardizing formats, and filling in missing information.

#### Key Steps:

- **Remove Duplicates:** Identify and eliminate any duplicate records that could distort analysis.
- **Fix Data Quality Issues:** Correct inconsistencies (e.g., formatting differences) and errors (like typos or outliers).
- **Handle Missing Data:** Decide how to address gaps. You can remove incomplete rows, fill in missing values (with averages, for example), or use imputation techniques.
- **Transform Data for Analysis:** Sometimes, data must be transformed into a suitable format (e.g., converting dates into standard formats or splitting text fields) for further processing.

**Why It Matters:** Clean, high-quality data leads to accurate, reliable analysis, reducing the risk of misleading conclusions.

## 4. Data Governance and Security

**What It Is:** This step involves setting policies and standards to manage data access, privacy, and security, ensuring that only authorized users can access specific data.

### Key Steps:

- **Define Access Controls:** Use role-based access to restrict data based on user roles, keeping sensitive information secure.
- **Set Privacy and Compliance Standards:** Data must comply with laws and regulations (e.g., GDPR, HIPAA) and should respect user privacy.
- **Create a Data Usage Policy:** Outline how data should be used, shared, and stored within the organization. Policies help prevent misuse and ensure data integrity.
- **Implement Security Protocols:** Use encryption, secure passwords, and regular audits to protect data against breaches.

**Why It Matters:** Governance and security ensure data is protected, usable, and compliant with legal standards, safeguarding both the business and its stakeholders.

## 5. Data Integration

**What It Is:** Data integration involves combining data from multiple sources into a cohesive, centralized format, enabling a comprehensive view of the business.

### Key Steps:

- **Establish Common Data Definitions:** Ensure that data fields are consistent across sources. For example, if “customer ID” appears in multiple databases, it should follow the same format.
- **Use ETL Tools:** Extract, Transform, Load (ETL) tools like Talend or Informatica facilitate data extraction, cleaning, and loading into a central repository.
- **Ensure Data Synchronization:** Data should be updated regularly across systems so that information remains accurate and up-to-date.
- **Resolve Data Conflicts:** Handle discrepancies (e.g., different names for the same customer across databases) to ensure that all data aligns.

**Why It Matters:** Integrated data gives a complete picture of business operations, supporting better analytics and decision-making by combining insights from multiple data sources.

## 6. Data Access and Analytics

**What It Is:** The final step is to make data accessible to users (often through dashboards or reports) so they can analyze and derive actionable insights.

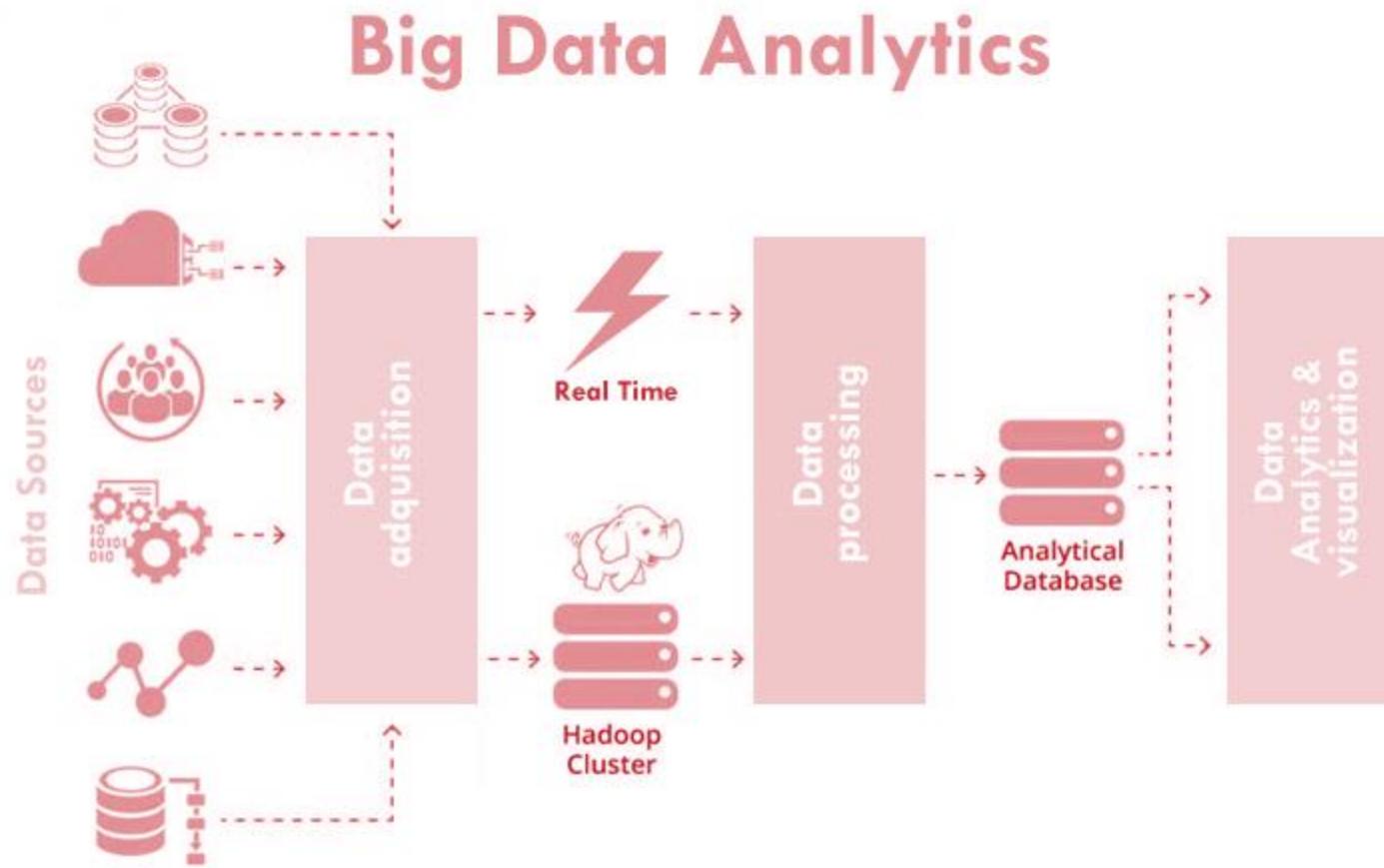
### Key Steps:

- **Implement Business Intelligence (BI) Tools:** Tools like Power BI, Tableau, or Looker visualize data for business users, making insights accessible and digestible.
- **Ensure Role-Based Data Access:** Only authorized users should access specific data, maintaining privacy and security.
- **Enable Self-Service Analytics:** Providing tools and resources for business users to perform their analyses can enhance decision-making across departments.
- **Measure Key Metrics and KPIs:** Define relevant metrics, such as customer retention rates or sales growth, to monitor performance and drive decisions.

**Why It Matters:** Proper storage ensures data is accessible, secure, and ready for analysis. It allows for efficient processing and retrieval, especially as the volume of data grows.

# Data management: basic concepts and fundamentals. DATA LIFECYCLE

## Data Life Cycle



## Example

### GEOTRACKING COMPANY:

**1. Devices in trucks send data to cloud:**

**2. App To Show Almost Real Time:**

- Truck/Drivers distance (Kms)
- Time in each status (driving, working, waiting & available)
- Documents

**3. ETL Pipeline to aggregate all collected valuable information**

**4. Data Storage: Data Warehouse for analytical purposes**

- Reporting services

## Deep review of the tool

- INFORMATION NEEDED
  - Distance
  - Speed
  - GPS
  - Brake
  - Fuel Consumption...
- INFORMATION AVAILABLE
  - Sensors:
    - Truck:
      - [CANBUS](#)
    - Trailer:
      - [CANBUS](#)
      - THERMOGRAPH



# Data management: basic concepts and fundamentals. DATA STORAGE

## Data Storage

1. Introduction to Data Storage
2. Relational Databases: SQL
3. No Relational Databases: NoSQL
4. Data Warehouses
5. Data Lakes
6. Case Study
7. Wrap-Up and Q&A

## Data Storage

### 1. Introduction to Data Storage

“Data is key to good decisions, but great data unlocks **powerful insights** that propel impactful actions”

Goal: efficient data storage for business insights

#### **Key Points:**

- There are different storage systems: databases, data warehouses, data lakes.
- Choosing the right storage for efficient data access and analysis.

**Discussion:** personal experiences with data storage (e.g., spreadsheets, cloud storage).

- **Bases de datos relacionales**

- Se utilizan en: aplicaciones tradicionales, DataWarehouse, ERP, CRM y e-commerce.
- Productos

## Data Storage

### 1. Relational Databases: SQL

Almacenan datos cuyas relaciones y esquema están predefinidos, diseñadas para admitir transacciones ACID y conservar la integridad referencial, así como la coherencia de los datos.

Database structure: tables, rows, columns, primary and foreign keys.

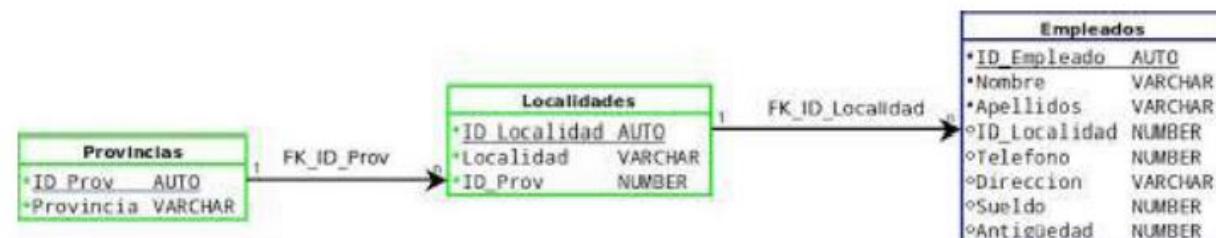
SQL basics: SELECT, INSERT, UPDATE, DELETE.

#### Products:

- MySQL, PostgreSQL, MariaDB, Oracle, SQL
- Amazon Aurora, Amazon RDS, Amazon

#### Assistance:

AI that helps to create a database:



**Discussion:** Simple SQL exercise on a sample database (e.g., employee or transportation data).

# Incapacidad de manejo de los datos del Método Tradicional

## Relational Database Management Systems (RDBMS)

- Terabytes y Petabytes de datos → no puede con ellos
  - Se necesitan cada vez Máquinas más potentes (más procesadores y memoria), y hacen poco viable su implementación
- 80% de los datos recopilados son semi-estructurados o no estructurados, con lo cual no puede analizarlos
- No puede lidiar con la velocidad de entrada de datos

## Data Mining vs Big Data

4 | 1 Introduction to the World of Big Data

Table 1.1 Differences in the attributes of big data and RDBMS.

| ATTRIBUTES    | RDBMS                  | BIG DATA                         |
|---------------|------------------------|----------------------------------|
| Data volume   | gigabytes to terabytes | petabytes to zettabytes          |
| Organization  | centralized            | distributed                      |
| Data type     | structured             | unstructured and semi-structured |
| Hardware type | high-end model         | commodity hardware               |
| Updates       | read/write many times  | write once, read many times      |
| Schema        | static                 | dynamic                          |

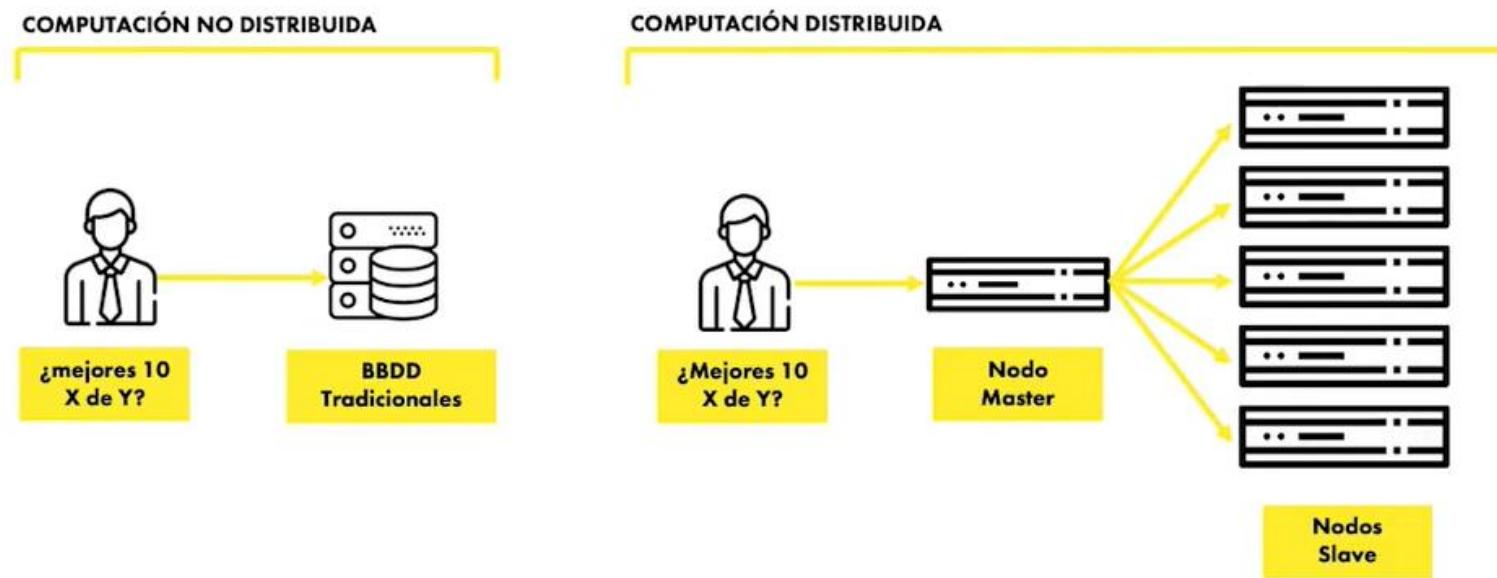
Table 1.2 Data Mining vs. Big Data.

| S. No. | Data mining                                                                                   | Big data                                                                                                                   |
|--------|-----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| 1)     | Data mining is the process of discovering the underlying knowledge from the data sets.        | Big data refers to massive volume of data characterized by volume, velocity, and variety.                                  |
| 2)     | Structured data retrieved from spread sheets, relational databases, etc.                      | Structured, unstructured, or semi-structured data retrieved from non-relational databases, such as NoSQL.                  |
| 3)     | Data mining is capable of processing large data sets, but the data processing costs are high. | Big data tools and technologies are capable of storing and processing large volumes of data at a comparatively lower cost. |
| 4)     | Data mining can process only data sets that range from gigabytes to terabytes.                | Big data technology is capable of storing and processing data that range from petabytes to zettabytes.                     |

# Distributed Storage

**Storage and processing of big volumes of data with speed and low cost.**

## DISTRIBUTED COMPUTING



## ¿Qué es BIG DATA?

### DISTRIBUTED STORAGE



# DATA STORAGE SOLUTIONS

## Most commonly used Big Data Storage Solutions

- Hadoop
- Elasticsearch
- Mongo db
- Hbase
- Cassandra
- Neo4J

Data management: basic concepts and fundamentals.  
**DATA CLEANING AND PREPARATION**

---

# Data Cleaning and Preparation

## Overview:

Often called "data wrangling," this process focuses on getting your data ready for analysis by correcting errors, harmonizing formats, and dealing with missing values.

## Key Processes:

- 1. Eliminate Duplicates:** Detect and remove repeated entries that could skew analytical results.
- 2. Resolve Data Quality Issues:** Address inconsistencies such as mismatched formats or errors, including typos and extreme outliers.
- 3. Address Missing Values:** Determine how to handle gaps in your data—options include removing incomplete entries, substituting missing values (e.g., with averages), or applying advanced imputation techniques.
- 4. Prepare Data for Analysis:** Modify and format data as needed, such as standardizing date formats, splitting combined text fields, or reorganizing datasets for easier analysis.

## Importance:

Accurate, well-prepared data ensures dependable insights and minimizes the risk of drawing incorrect conclusions.

# Data Cleaning and Preparation

**Beyond Basics:** Once the foundational cleaning is done, advanced techniques can further optimize your data for analysis and enhance the quality of your insights.

## Key Advanced Steps:

### 1. Feature Engineering:

1. Create new variables or modify existing ones to uncover additional insights.
2. Example: Deriving age from a date of birth or calculating a profitability ratio.

### 2. Normalization and Scaling:

1. Adjust numerical data to a common scale without distorting relationships.
2. Example: Scaling income data to fall between 0 and 1 for machine learning models.

### 3. Outlier Treatment:

1. Use statistical methods to identify and handle outliers that could bias results.
2. Approaches: Winsorization, clipping, or applying robust statistical techniques.

### 4. Data Enrichment:

1. Integrate additional data sources to provide more context.
2. Example: Augmenting sales data with weather information for trend analysis.

### 5. Automating the Process:

1. Leverage tools or scripts (e.g., Python, R) to automate repetitive cleaning and preparation tasks, improving efficiency and consistency.

**Why It's Critical:** Advanced preparation ensures your data is not just clean but also tailored to the analytical techniques you plan to use, enabling deeper insights and more effective decision-making.

## Data Cleaning and Preparation

### STEPS:

1. WHAT DO WE WANT? WHY WE ARE DOING THIS ANALYSIS?
2. WHAT DATA DO WE HAVE?

- UNDERSTANDING DATA
  - A. STRUCTURED AND SEMISTRUCTURED DATA
  - B. UNSTRUCTURED DATA

A: FIRST STEPS ARE GOING TO BE PERFORMED WITH AN ETL TOOL THAT WE ALL HAVE IN OUR COMPUTERS BUT NEVER NEW WE HAD IT.

We are going to dive in a tool that everybody has in its computer: POWERQUERY FOR EXCEL.

What we are going to learn is also valid for a business intelligence tool like POWER BI.

# Data Cleaning and Preparation

## STEPS:

1. What do we want? Why we are doing this analysis?
2. What data do we have?
  - Understanding data
    - A. Structured and semistructured data
    - B. Unstructured data
3. Beyond the basics: outlier treatment, new variables, enrich data
4. Automate: scripts r/Python
5. Connecting those scripts with Workflows or cronjobs/scheduled tasks

Data management: basic concepts and fundamentals.  
**DATA GOVERNANCE AND SECURITY**

---

## 4. DATA GOVERNANCE AND SECURITY

**Establishing policies for data access, privacy, and compliance** ensures sensitive information is protected and regulatory requirements are consistently met. This foundation supports trustworthy, ethical, and efficient use of data across the organization.

- **ACCESS CONTROL:** Defines clear permissions and roles to ensure that only authorized individuals can access specific datasets, minimizing the risk of unauthorized use or breaches.
- **PRIVACY:** Implements robust measures to safeguard personal and sensitive information, adhering to frameworks such as GDPR, HIPAA, or CCPA to build trust with stakeholders.
- **COMPLIANCE:** Aligns data management practices with legal and industry regulations, ensuring that data usage meets all standards for security, auditability, and accountability.
- **TRANSPARENCY:** Creates clear documentation and audit trails for how data is collected, stored, shared, and processed, promoting organizational integrity and readiness for regulatory audits.
- **RISK MANAGEMENT:** Proactively identifies vulnerabilities and establishes protocols to prevent, detect, and respond to potential data breaches or misuse.

Through strong governance and security practices, organizations can not only protect their data assets but also foster a culture of responsibility, mitigate risks, and enhance the reliability of their data-driven initiatives.

# Data management: basic concepts and fundamentals. DATA INTEGRATION

## 5. DATA INTEGRATION

**Combining data from multiple sources**, such as CRM systems, marketing platforms, or financial databases, creates a unified and holistic view of organizational information. This enables seamless analysis, improved collaboration, and more strategic decision-making.

- **CONNECTIVITY:** Ensures smooth and reliable access to various data sources, facilitating real-time or scheduled synchronization to keep information up-to-date across systems.
- **TRANSFORMATION:** Standardizes, cleanses, and enriches data during integration, ensuring consistency, accuracy, and usability for downstream analytics and reporting.
- **VISIBILITY:** Provides stakeholders with a comprehensive, 360-degree view of operations, customers, and performance by breaking down silos between disparate data systems.
- **EFFICIENCY:** Automates data workflows and reduces manual effort, accelerating the availability of integrated data for timely insights and action.

By enabling the seamless merging of data, integration supports scalable analytics, drives operational excellence, and aligns cross-functional objectives, all while ensuring compliance and governance of data assets.

# Data management: basic concepts and fundamentals. DATA ACCESS AND ANALYTICS

## 6. DATA ACCESS AND ANALYTICS

**Making data accessible to the right people at the right time** ensures that stakeholders can act on accurate, timely information, often through dashboards, reports, or analytics tools, fostering a culture of data-driven decision-making across the organization.

- **NOTIFICATIONS:** Serve as proactive alerts, ensuring that users are immediately informed of significant changes, anomalies, or opportunities within the data, enabling swift action and reducing response times.
- **REPORTS:** Provide structured, in-depth insights with formulas, graphs, and data visualizations, summarizing historical trends and key performance metrics to inform strategy and operations.
- **DASHBOARD:** Interactive dashboards consolidate and visualize data in real-time, making complex information more accessible and actionable. By bringing businesses closer to their data, dashboards empower users to monitor performance, identify patterns, and make informed decisions with agility.

This approach to data access and analytics aligns with broader data management goals, including ensuring data integrity, compliance, and scalability to support evolving business needs.

## 02. Introduction to massive data processing: infrastructures, types, development and applications

1. INTRODUCTION TO BIG DATA INFRASTRUCTURES
2. DISTRIBUTED COMPUTING
3. HADOOP ECOSYSTEM
4. DATA PROCESSING: BATCH & STREAM PROCESSING
5. CLOUD COMPUTING FOR BIG DATA

## 1. INTRODUCTION TO BIG DATA INFRASTRUCTURES

### **HDFS (Hadoop Distributed File System)**

#### **Definition:**

HDFS is a distributed file system.

It combines the storage capacity of multiple distributed machines into a single, very large system.

It allows creating directories and storing data in files in a distributed manner across multiple machines.

#### **Key Features of HDFS:**

- **Distributed:**  
Data is stored across multiple machines, making it scalable and fault-tolerant.
- **Horizontally Scalable:**  
New machines can be added to the system to increase storage and processing capacity.
- **Cost-Effective:**  
Allows starting with simple, low-cost machines and scaling up as needed.
- **Fault Tolerant:**  
Designed to handle hardware failures without data loss or downtime.
- **High Throughput:**  
Optimized for large-scale data processing, ensuring efficient performance.

# 1. INTRODUCTION TO BIG DATA INFRASTRUCTURES

Explosion in the amount of data generated by:

- Applications: Social networks, Web applications, Mobile apps, IoT devices, Log files, etc.

It is necessary to explore methods that enable:

## 1. Transmitting and distributing data between distributed applications

(Message brokers: RabbitMQ, ActiveMQ, Kafka)

## 2. Storing and securing data in a distributed manner

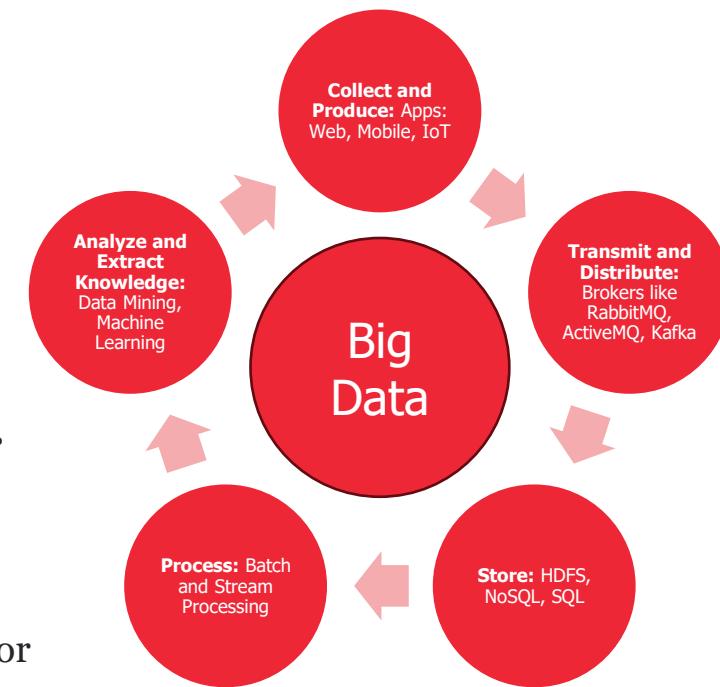
(Technologies: Hadoop HDFS, NoSQL databases such as Cassandra, MongoDB, HBase, Elasticsearch, etc.)

## 3. Processing and analyzing data in a distributed way to extract knowledge for decision-making

1. Big Data Processing: Batch Processing (e.g., MapReduce, Spark) and Stream Processing (e.g., Kafka Stream, Flink, Storm, Samza)
2. Data Mining and Machine Learning: TensorFlow, DeepLearning4J, Weka, etc., for knowledge extraction.

## 4. Analyzing and visualizing decision-making indicators

(Using Big Data visualization tools)



# 1. INTRODUCTION TO BIG DATA INFRASTRUCTURES

## Big Data: The 3 V's

### 1. Volume

1. The quantity of data (Measured in Petabytes - PB)

### 2. Variety

1. Different formats of data (Structured: 20%, Unstructured: 80%)
  1. Examples: Text, CSV, XML, JSON, Binary files, Databases, etc.

### 3. Velocity

1. The frequency at which data is generated
  1. Example: Twitter
    1. Every second, approximately 5,900 tweets are sent on the Twitter microblogging platform.
    2. This represents 504 million tweets per day or 184 billion tweets per year.
    3. This mass of information contributes to the stream of "Big Data" published by humanity daily on the internet.

# 1. INTRODUCTION TO BIG DATA INFRASTRUCTURES

## Big Data: 5 V's

### 1. Volume

1. The amount of data.

### 2. Variety

1. The diversity of data formats and types.

### 3. Velocity

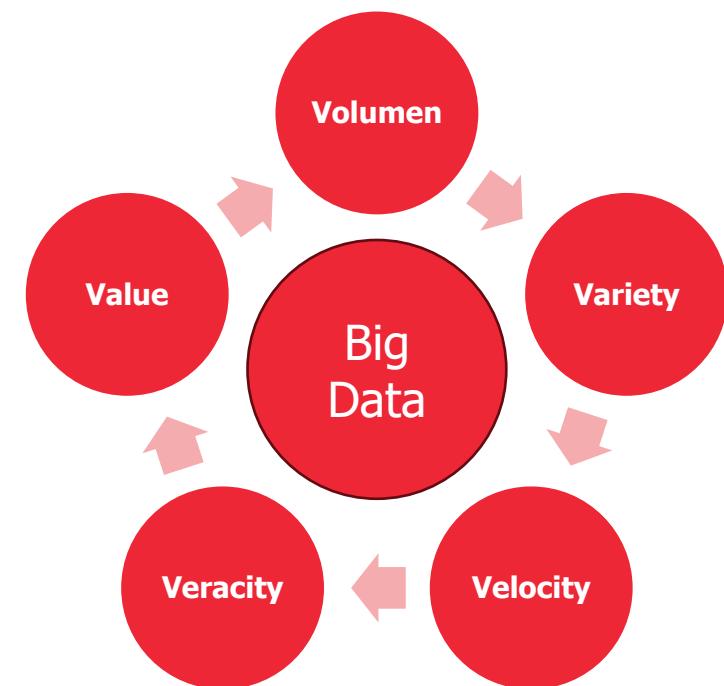
1. The speed at which data is generated and arrives.

### 4. Veracity

1. Reliability and credibility of the collected data (from reliable sources).

### 5. Value

1. The profit and knowledge that can be extracted from the data.
2. Transforming data into actionable value.



# 1. INTRODUCTION TO BIG DATA INFRASTRUCTURES

## EXAMPLE OF PROBLEM: TRANSPORT COMPANY

### 1. Volume

The company generates a vast amount of data from its operations:

- GPS tracking data from thousands of trucks.
- Fuel consumption records.
- Delivery schedules for hundreds of customers daily.
- Sensor data (temperature, humidity) for perishable goods.
- Electronic Proof of Delivery (ePOD) and invoices.

### 2. Variety

The data comes in diverse formats:

- **Structured Data:** Customer orders, delivery schedules, financial transactions.
- **Semi-Structured Data:** GPS logs, IoT sensor readings, and email communications.
- **Unstructured Data:** Customer feedback, driver reports, and images of damaged goods.

### 3. Velocity

The speed at which the data is generated and needs processing is crucial:

- Real-time GPS updates (every second) to track trucks' locations.
- Continuous temperature monitoring for refrigerated goods.
- Real-time alerts for delivery delays, accidents, or maintenance issues.
- Daily route optimization based on traffic and weather data.

### 4. Veracity

Ensuring data accuracy and reliability is critical:

- Verifying sensor data for accurate temperature readings in cold-chain logistics.
- Ensuring customer delivery addresses are correct to avoid delays.
- Filtering out inaccurate GPS data caused by signal loss.
- Using reliable data sources (e.g., government traffic databases) to avoid false routing decisions.

### 5. Value

Extracting actionable insights from the data to drive profitability:

- **Optimization:** Use Big Data to reduce fuel costs by optimizing routes and eliminating empty miles.
- **Customer Satisfaction:** Predict delivery times more accurately, improving customer trust and loyalty.
- **Predictive Maintenance:** Use IoT data from trucks to predict and prevent breakdowns, minimizing downtime.
- **Strategic Decisions:** Analyze demand trends to reposition trucks more effectively, reducing idle time.

# DISTRIBUTED COMPUTING

## Big Data in Everyday Life: A Chef Processing Orders

This analogy explains Big Data processing using the example of a chef handling orders in a restaurant:

### 1.Orders :

1. Just like a restaurant receives multiple orders from customers, Big Data systems collect large amounts of incoming data from various sources.
2. **Example:** Online orders, reservation data, or customer preferences.

### 2.Job:

1. Each order represents a "job" that needs to be completed.
2. The chef (Big Data system) processes these jobs based on priorities, resources, and capacity.
3. **Example:** The chef prioritizes meals based on preparation time and delivery requirements.

### 3.Processing:

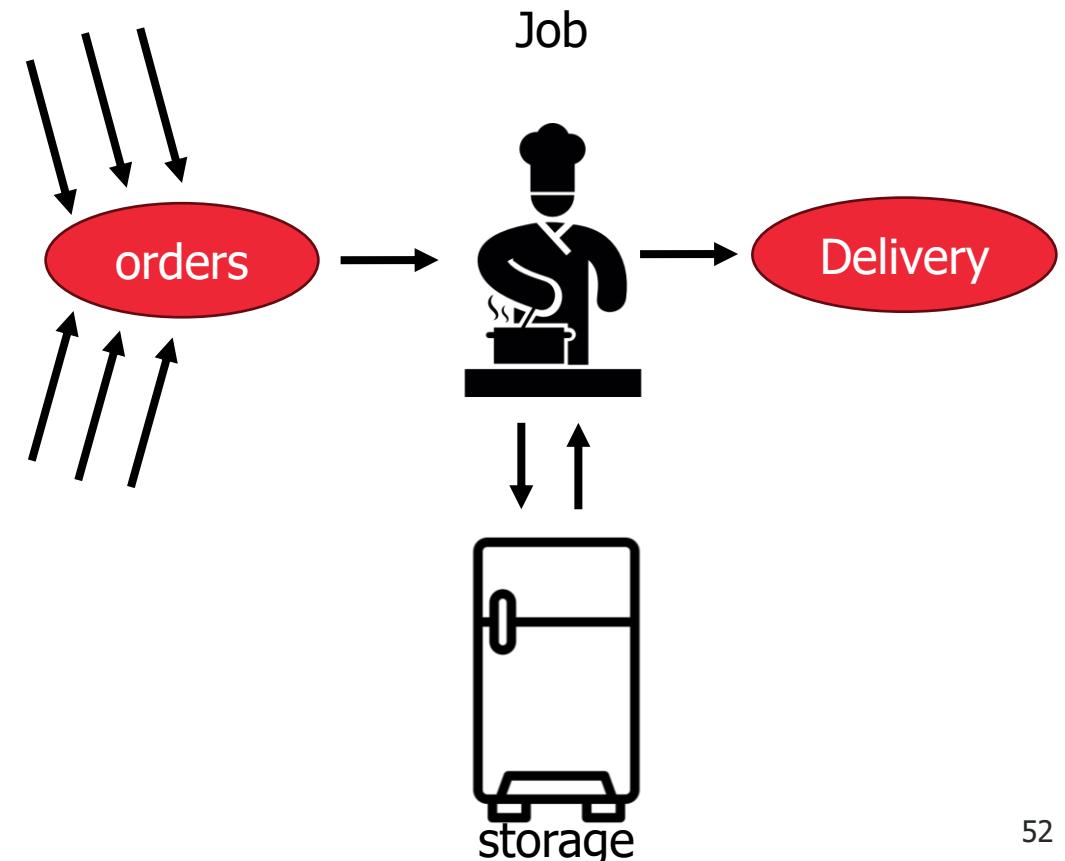
1. The chef processes the orders by cooking the meals, akin to Big Data systems processing raw data into actionable insights.
2. **Example:** In Big Data, this might involve analyzing customer trends or optimizing operations.

### 4.Storage:

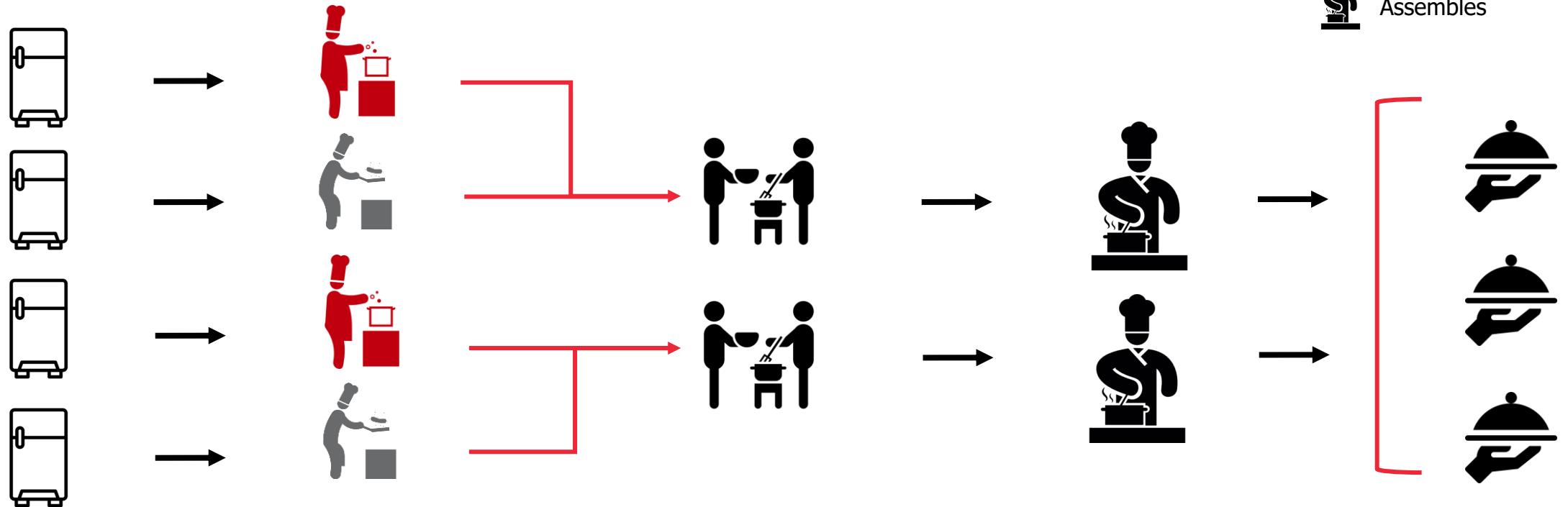
1. Just like the chef uses a refrigerator to store ingredients for future use, Big Data systems store processed or raw data for later analysis.
2. **Example:** Data stored in databases (HDFS, NoSQL) or cold storage for long-term archiving.

### 5.Delivery:

1. After processing, the chef delivers the meals to customers. Similarly, Big Data delivers results (insights) to end-users or decision-makers.
2. Examples: Reports, dashboards, or notifications sent to stakeholders



## DISTRIBUTED COMPUTING



### 3. BIG DATA ECOSYSTEM – APACHE HADOOP

Hadoop is a free and open-source framework written in Java, designed to facilitate the creation of massively distributed applications across thousands of nodes.

- **At the storage level:** Distributed storage of data (petabytes of data) with **HDFS** (Hadoop Distributed File System).

- **At the processing level:** Data processing with **MapReduce**.

- **Supports fundamental features:**

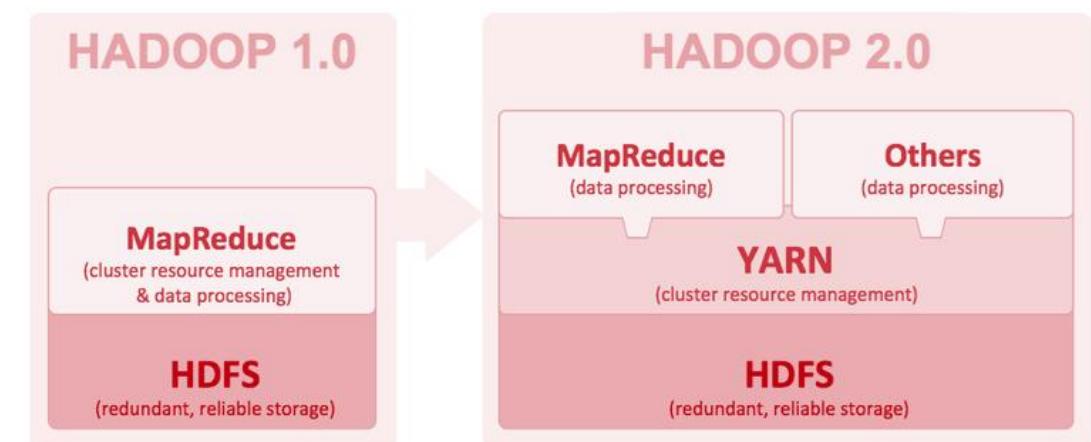
- High availability
- Scalability
- Fault tolerance
- Recovery after failure
- Security (HPC: High-Performance Computing)

The core Hadoop framework is composed of the following modules:

- **Hadoop Distributed File System (HDFS):** A system for distributed file storage.

- **Hadoop YARN (Yet Another Resource Negotiator):** A system for cluster resource management.

- **Hadoop MapReduce:** Distributed data processing.



### 3. HADOOP ECOSYSTEM:

DATA MANAGEMENT

DATA ACCESS

DATA PROCESSING

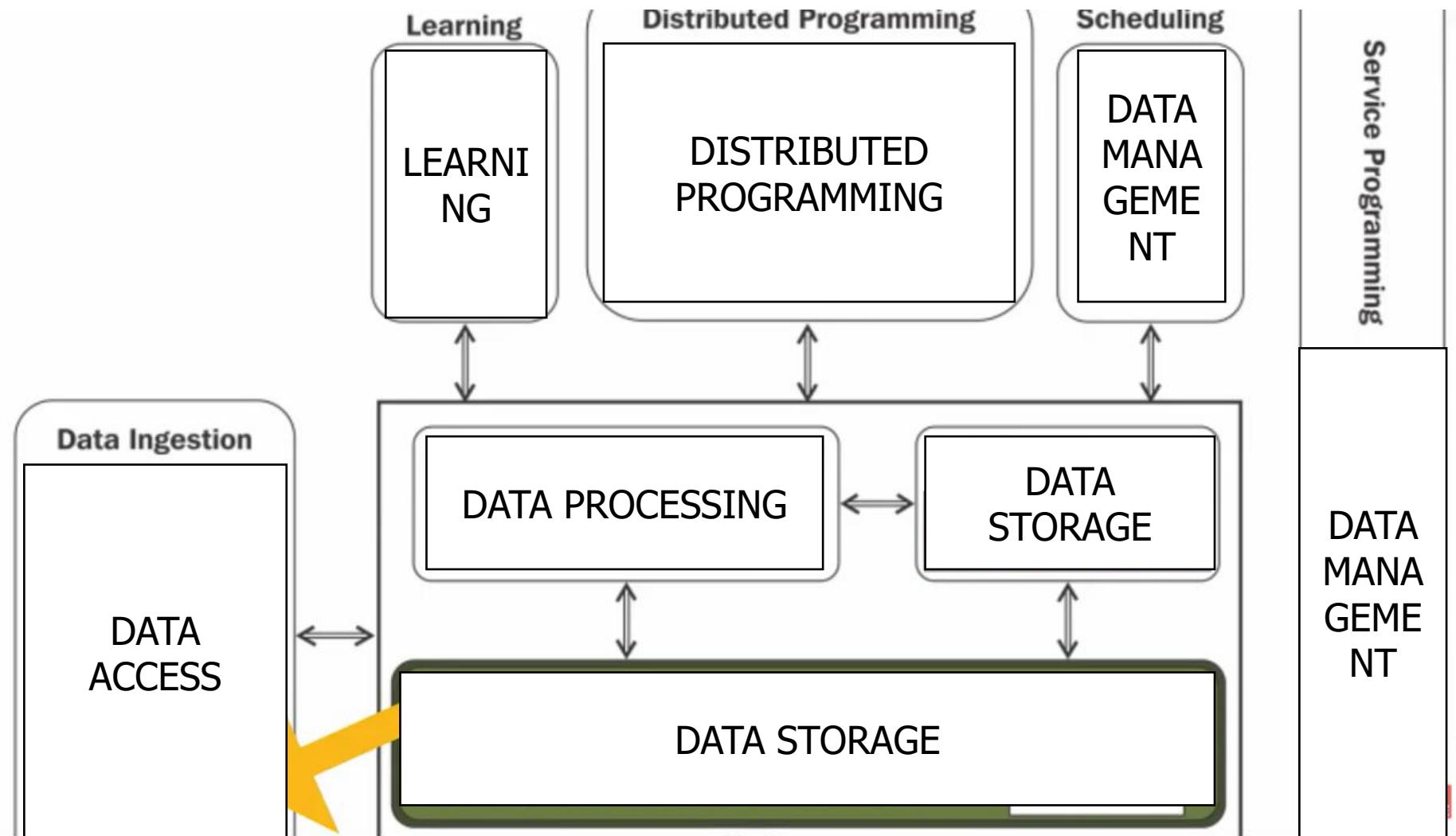
DATA STORAGE

DISTRIBUTED PROGRAMMING

DISTRIBUTED LEARNING/SCHEDULING

### 3. HADOOP ECOSYSTEM:

HDFS  
Hadoop MapReduce  
Apache Hbase  
Apache Spark  
Apache Hive  
Oozie  
Mahout  
Sqoop  
Storm  
Flume



## Enumeración de las más habituales:

- Ecosistema Hadoop

**1-HDFS**

**2-Hadoop MapReduce**

**3-Apache Hbase**

**4-Apache Spark**

**5-Apache Hive**

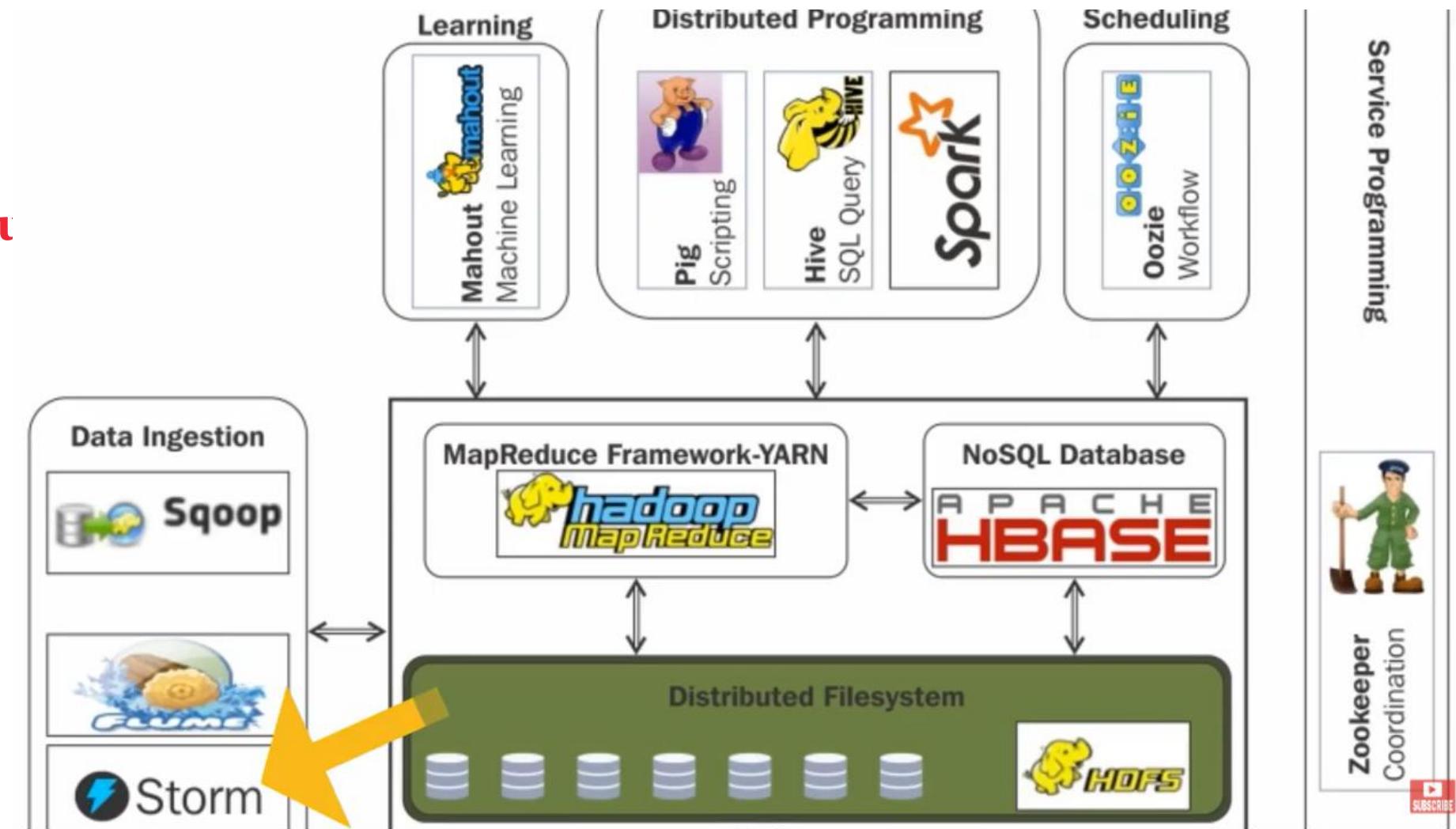
**6-Oozie**

**7-Mahout**

**8-Sqoop**

**9-Storm**

**10-Flume**



## 4. DATA PROCESSING – BATCH AND STREAM

### Batch Processing (Traitement par lots):

#### • Definition:

- Batch processing refers to the processing of blocks of already stored data over a specific period.

#### • Example:

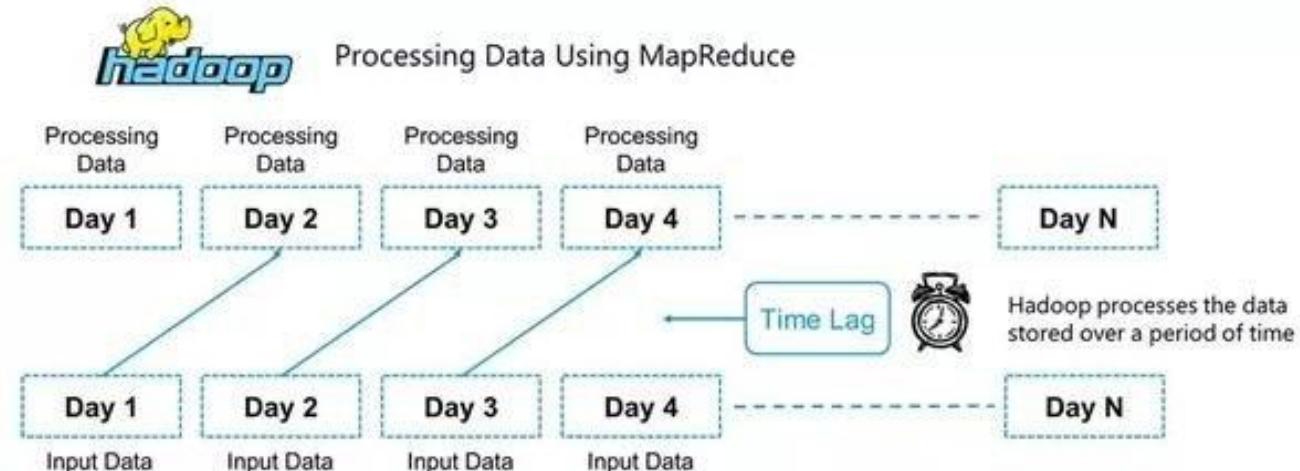
- Processing all the transactions performed by a financial company over the span of a week.

#### • Characteristics:

- The data contains millions of records for each day. These records can be stored as:
  - Text files (e.g., CSV format).
  - Records stored in systems like HDFS, SQL databases, or NoSQL databases.

#### • Framework Examples:

- **MapReduce** (Hadoop): A framework for batch data processing.
- **Spark**: Another framework designed to process large batches of data efficiently.

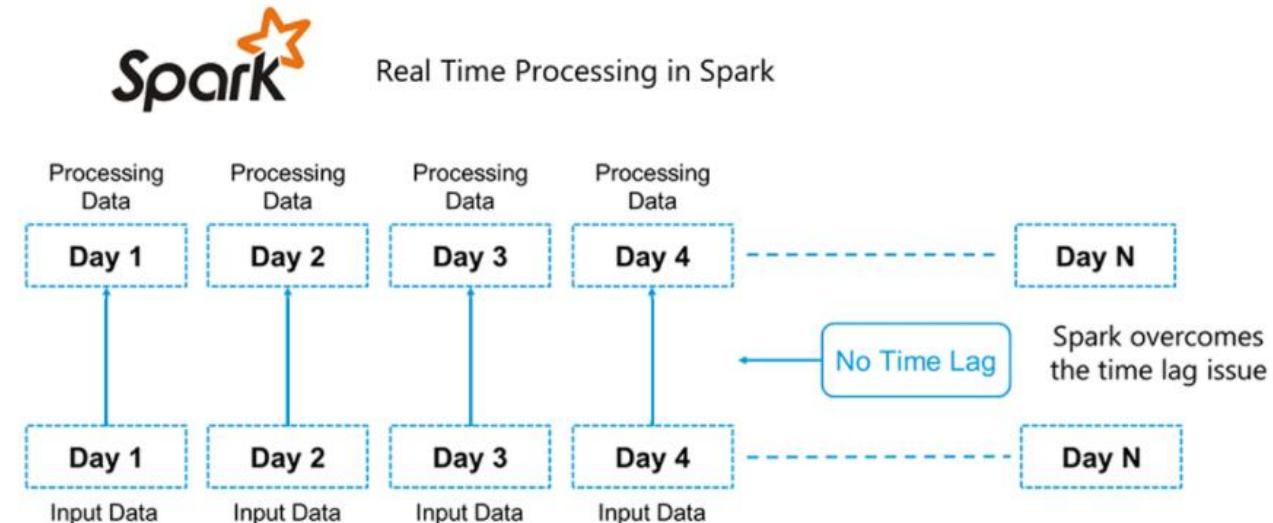


## 4. DATA PROCESSING – BATCH AND STREAM

### Stream Processing (Real-Time Processing):

- Unlike batch processing, where data has a defined start and end and is processed once all the data is available, stream processing is designed for continuously flowing data in real time, arriving endlessly for days, months, years, or indefinitely.
- Stream processing allows:

- **Real-time data processing:** Data can be handled as soon as it is generated.
- **Instant analysis results:** Stream processing feeds data directly into analytics tools, enabling immediate insights.



## 4. DATA PROCESSING – BATCH AND STREAM

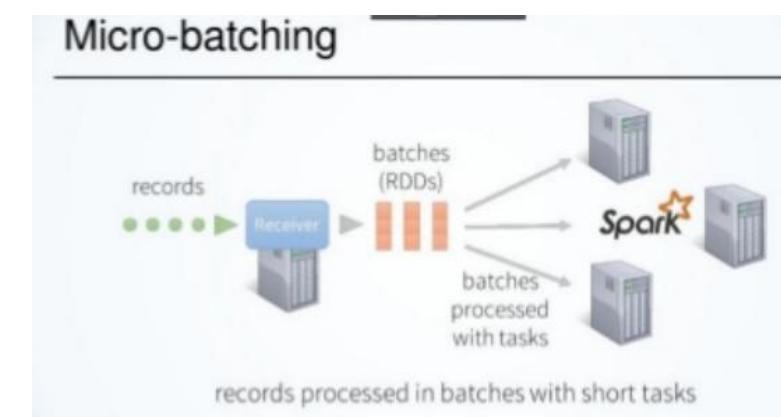
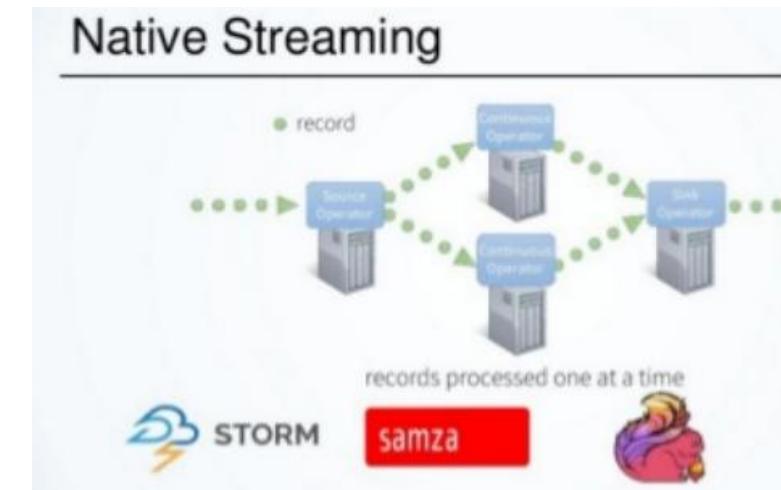
### Stream Processing Approaches

#### 1. Native Streaming (Real-Time Processing):

- **Definition:**
  - Each incoming record is processed immediately as it arrives, without waiting for other data.
- **Examples:**
  - Technologies: **Storm, Flink, Kafka Streams, Samza.**

#### 2. Micro-Batch Processing:

- **Definition:**
  - Incoming records over short intervals (e.g., seconds) are grouped into small batches and processed together with minimal delay.
- **Examples:**
  - Technologies: **Spark Streaming, Storm-Trident.**



## 4. DATA PROCESSING – STREAM PROCESSING – STRONG AND WEEK POINTS BTW NATIVE STREAMING AND MICRO-BATCH PROCESSING

### Native Streaming:

#### Strong Points:

- **Real-Time Results:** Immediate processing makes it ideal for applications requiring instant insights (e.g., fraud detection, live monitoring).
- **Low Latency:** Data is processed as soon as it arrives, ensuring minimal delays.
- **Granular Processing:** Handles each event independently, which is useful for real-time applications.

#### Weak Points:

- **Complexity:** Requires more sophisticated programming models for event-by-event processing.
- **Resource Intensive:** Constant processing can strain system resources.
- **Fault Tolerance:** Ensuring data consistency in case of failures is more challenging.

### Micro-Batch Processing:

#### Strong Points:

- **Efficiency:** Processing small batches reduces overhead compared to handling individual events.
- **Simpler Model:** Easier to implement and manage, as it follows a more structured batch-like approach.
- **Integration:** Compatible with existing batch processing systems (e.g., Hadoop) for hybrid workflows.

#### Weak Points:

- **Latency:** Slight delays are introduced due to batch intervals, making it less suitable for strict real-time requirements.
- **Granularity:** Lacks the fine-grained processing of native streaming, as events are grouped into batches.

## 4. DATA PROCESSING – STREAM PROCESSING – CHOOSING THE RIGHT APPROACH

### Choosing the Right Approach

- **Native Streaming:** Best for applications that demand immediate action, such as financial fraud detection or live stock market analysis.
- **Micro-Batch Processing:** Suitable for near-real-time analytics and applications where small delays are acceptable, such as dashboard updates or periodic reports.

## 4. BIG DATA ECOSYSTEM ...KEEPS EXPANDING

The Big Data ecosystem has expanded with many tools, such as:

### Stream Processing:

- **Apache Storm:** A framework for computation and distributed processing of data streams (Stream Processing).
- **Apache Flink:** A framework for computation and distributed processing of data streams (Stream Processing).
- **Apache Spark:** A framework for distributed Big Data processing, offering an alternative to MapReduce and supporting Stream Processing.
- **Apache Kafka Streams:** A real-time streaming platform for distributed applications and message systems.

**Apache Zookeeper:** A system for managing the configuration of distributed systems to ensure coordination between nodes.

### NoSQL Databases (SQBD NoSQL):

- **Apache HBase:** A distributed NoSQL database for structured storage in large tables. → NEEDS HDFS!!
- **MongoDB:** A NoSQL database (Not Only SQL) where data is stored in a distributed manner across nodes in JSON document format.
- **Cassandra:** A NoSQL database (Not Only SQL) where data is stored in a distributed manner across nodes in JSON document format.
- **Elasticsearch:** A distributed and multi-entity search engine accessible through a REST interface.

**Hazelcast:** A distributed in-memory cache and NoSQL system, also functioning as a message application system.

## 4. BIG DATA ECOSYSTEM ...KEEPS EXPANDING

The Big Data ecosystem has expanded with many tools, such as:

- **Apache Pig:**  
A high-level platform for creating MapReduce applications using the Pig Latin language (which resembles SQL) instead of writing Java code.
- **Apache Hive:**  
A data warehouse infrastructure for analysis and querying using a language similar to SQL.
- **Apache Phoenix:**  
A relational database engine built on top of HBase.
- **Apache Impala:**  
A SQL query engine from Cloudera for systems based on HDFS and HBase.
- **Apache Flume:**  
A system for collecting and analyzing log files.
- **Apache Sqoop:**  
Command-line tools for transferring data between relational databases (RDBMS) and Hadoop.
- **Apache Oozie:**  
A workflow scheduling tool for managing Hadoop data processing workflows.

## 5. INTRODUCTION FOR CLOUD COMPUTING IN BIG DATA

Cloud computing provides on-demand access to scalable computing resources, including storage, compute power, and applications.

### 1. Why the Cloud for Big Data?

Scalability, Cost Effectiveness, Flexibility, Global Access

### 2. Key Cloud Service Models for Big Data:

- Infrastructure as a Service (IaaS):
- Platform as a Service (PaaS):
- Software as a Service (SaaS):

### 3. Benefits of Cloud Computing for Big Data

Elastic Scalability, Cost Optimization, Built-In Tools, Reduced Time to Market: Disaster Recovery:

### 4. Cloud Platforms for Big Data

AWS, Azure, GCP, IBM, Oracle, Cloudera

### 5. Tools and Frameworks for Big Data in the Cloud

Data storage, Data Processing and Data Integration

### 6. Real-World Use Cases

Social Media Analysis, IoT Data Processing, E-commerce Analytics

## 5. INTRODUCTION FOR CLOUD COMPUTING IN BIG DATA

### 7. Challenges and Considerations

- Data Security and Privacy: Highlight concerns about storing sensitive data in the cloud. Discuss encryption and compliance solutions offered by providers.
- Cost Management: warn about potential overuse of resources without proper cost monitoring tools.
- Data Transfer: discuss potential latency and costs associated with moving large datasets to/from the cloud

### 8. Future Trends:

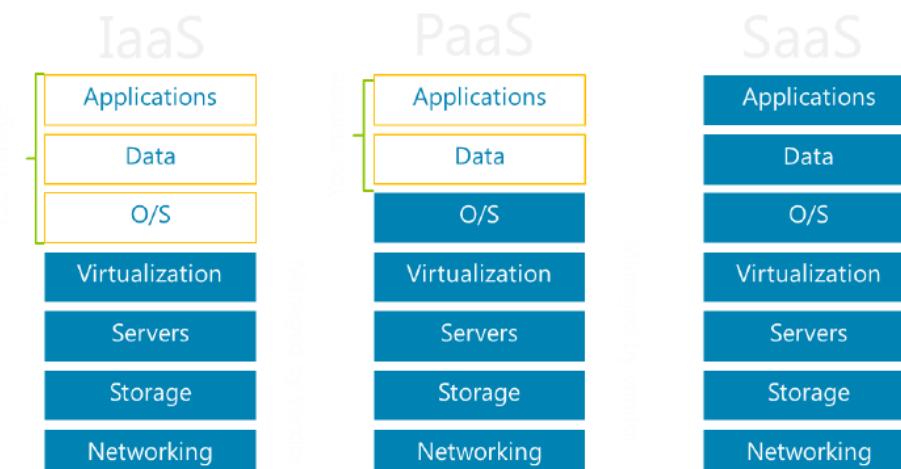
- Infrastructure as a Service (IaaS):
- Serverless Computing: serverless architectures (e.g., AWS Lambda, Azure Functions) simplify Big Data workloads by abstracting infrastructure.
- AI and Big Data Integration: Cloud-based AI/ML tools like Google AI Platform and AWS SageMaker.
- Multi-Cloud and Hybrid Models: Using multiple cloud providers or a combination of on-premises and cloud for flexibility.

# Opciones de Servicios en Entornos Cloud

## Entorno Cloud. ¿Decisiones arquitecturales, financieras, de negocio?

- **Estrategia fiscal:**
  - CAPEX (Capital Expenditure) vs
  - OPEX: Operational Expenditure
- **Aspectos financieros:**
  - Total Cost of Acquisition (TCA) vs
  - Total Cost of Ownership (TCO)
- Time To Market / Revenue

¿Qué modalidad de Cloud?



IaaS



PaaS



SaaS

## 03. Application deployment. Development of scalable applications

---

## How to deploy this applications:

1. Manual installations: install Hadoop and Spark ([Link1](#)) y [multinode](#)
2. Could computing distributions: services used where installation and scalability is simpler and easily scalable.
  1. Cloudera
  2. Hortonworks
  3. [Google](#)
  4. AWS
  5. MICROSOFT
  6. IBM

Best things are the scalability. You can create them for a particular use case and when you want deploy it and increase the size as you need.

When you stop needing it, you can erase it and you Will not be charged for any of it.

## How to deploy this applications:

1. What do I need to know to install manually?
  - a. basic cmd/bash commands
  - b. download the software needed
    - Desired Software
    - Dependencies needed: for example Java
  - c. setup properly the machine
    - I. Environmental variables
    - II. Networking: ip addresses
    - III. communication protocols: ssh (keys for security encryptions)
  - d. install and verify prerequisites (verify versions with commands like: java --version, python3 --version, pip..., system utilities: curl, wget, unzip ...)
  - e. Install the Big Data Tool
    - a. Follow the installation guide for the software
      - a. extract downloaded binaries
      - b. move to a suitable directory
      - c. Run anysetup scripts provided
    - b. Create system services for tools that run continuously (Linux .service in etc/systmemd/system, Windows use task Scheduler)

## How to deploy this applications:

1. What do I need to know to install manually?
  - f. Configure the Tool
    - i. Modify configuration files as required:
      - core-site.xml, hdfs-site.xml, and mapred-site.xml for Hadoop.
      - spark-defaults.conf, spark-env.sh for Spark.
    - ii. Set appropriate resource limits: Memory allocation Number of threads
    - iii. Enable logging:
      - Specify log levels (INFO, DEBUG, ERROR)
      - Configure log rotation
  - g. Validate the Installation
    - i. Run basic commands to test functionality:
      - Hadoop: hadoop version, hdfs dfs -ls /
      - Spark: spark-submit --versión
    - ii. Verify services are running:
      - Check processes with ps or tasklist.
      - Use web UIs (e.g., Hadoop ResourceManager, Spark Master).

## How to deploy this applications:

1. What do I need to know to install manually?
  - h. Optimize for Performance
    - i. Adjust system limits (ulimit, kernel parameters).
    - ii. Optimize JVM settings for Java-based tools:
      - Example: -Xms2g -Xmx4g
    - iii. Enable caching and compression for large data handling
  - i. Set Up Monitoring
    - i. Install monitoring tools:
      - JMX Exporter, Prometheus, Grafana for metrics.
      - Log analyzers like Elasticsearch and Kibana.
    - ii. Monitor system resources:
      - Disk usage, CPU, memory, network throughput.
  - j. Document the Process
    - i. Keep a log of installation steps and configurations for future reference.
    - ii. Create a backup of configuration files.

# Deploy Spark

## **Option 1: Virtual machine (in a Linux OS)**

- **Steps:**
  - **Install VM Virtualbox and Ubuntu**
  - **Follow this steps in this video to install spark: [link](#)**

## **Option 2:**

- using docker: pyspark jupyterlab -- run it with a command:

```
docker run -p 8888:8888 -p 4040:4040 -e JUPYTER_ENABLE_LAB=yes -v  
/c/Users/marti/OneDrive/Documentos/Notebooks:/home/jovyan/work jupyter/pyspark-notebook
```

## Deploy AirFlow

### **Option 1: Virtual machine (in a Linux OS)**

- **Steps:**
  - *Install VM Virtualbox and Ubuntu*
  - *Follow this steps in this video to install spark: [link](#)*

### **Option 2:**

- using docker

### **Option 3:**

- docker compose

# Applications Deployment: Development of Scalable Applications for Big Data Processing

# Development of Scalable Applications for Big Data Processing

## Designing for High-Volume, High-Velocity & High Variety Data

Big Data Applications must handle high velocity (streaming), massive volumes, and diverse data.

Example: Netflix processes petabytes daily for recommendations and analytics.

## Why Scalability Matters?

Real-time analytics, efficient storage, and seamless scaling.

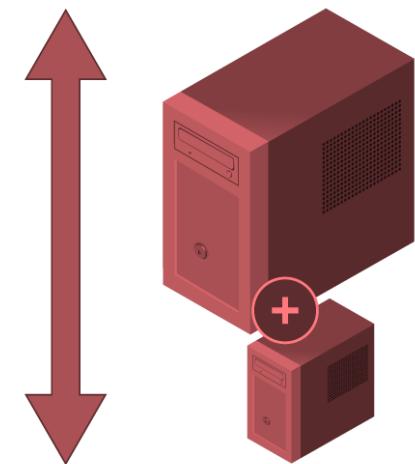
Why is critical in Big Data?

- Manage exponential data growth
- Enable real-time insights for streaming data
- Quickly adapt to new data sources and frameworks
- Ensure high availability and fault tolerance

## Development of Scalable Applications for Big Data Processing:

### SCALING:

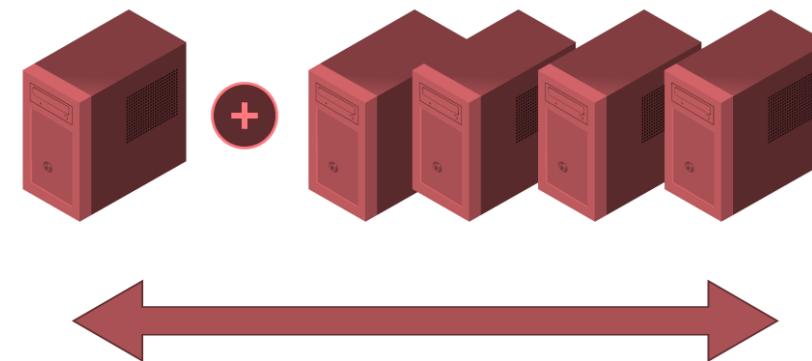
#### VERTICAL vs. HORIZONTAL



Vertical Scaling  
Up and Down

#### UPGRADING

RAM/CPU/STORAGE



Horizontal Scaling  
Out and In

ADDING MORE NODES

## Development of Scalable Applications for Big Data Processing:

### CHALLENGES:

- Managing distributed systems.
- Sharding and partitioning data across nodes.
- Ensuring fault tolerance.
- Balancing compute vs. storage requirements.
- Handling varying workloads effectively.

### BEST PRACTICES FOR BIG DATA SCALABILITY:

- Use distributed frameworks like Apache Spark, Kafka, or Flink.
- Leverage cloud-based storage (e.g., Amazon S3, Google BigQuery).
- Optimize data pipelines for streaming frameworks.
- Design for modularity and scalability in architecture.

# Development of Scalable Applications for Big Data Processing:

## TOOLS & TECHNICS

Tools and Techniques for Big Data Scalability

### ***1. Distributed Storage***

HDFS (Hadoop Distributed File System): A reliable and scalable storage system for managing large datasets.

Amazon S3: Cloud-based object storage with high scalability and durability.

Google Cloud Storage: A highly available, secure, and scalable storage option for big data processing.

### ***2. Data Processing Frameworks***

Apache Hadoop: Batch processing for large-scale data analysis.

Apache Spark: Unified analytics engine for large-scale data processing with in-memory computation.

Apache Flink: Stream and batch processing framework optimized for real-time analytics.

### ***3. Streaming Tools***

Apache Kafka: Distributed event streaming platform for real-time data pipelines.

AWS Kinesis: Scalable and fully managed platform for streaming data processing.

Apache Pulsar: Multi-tenant, high-performance message broker for streaming and queuing.

# Development of Scalable Applications for Big Data Processing:

## TOOLS & TECHNICS:

### ***4. Workflow Orchestration and Automation***

Apache Airflow: A platform to programmatically author, schedule, and monitor workflows.

Apache NiFi: Data flow automation for system integration, transformation, and routing.

Luigi: Python-based workflow management system designed for batch processes.

Prefect: Modern workflow orchestration platform with an emphasis on simplicity and reliability.

### ***5. Monitoring and Optimization***

Prometheus: Open-source monitoring and alerting toolkit for system metrics.

Grafana: Visualization and analytics software for monitoring dashboards.

Datadog: Cloud-based monitoring and performance tracking for infrastructure and apps.

Elastic Stack (ELK): Log management and analytics platform for monitoring and troubleshooting.

# Development of Scalable Applications for Big Data Processing:

## TOOLS & TECHNICS

### ***6. ETL and Data Integration***

Talend: Data integration platform for transforming, cleansing, and loading data.

Informatica: Enterprise-grade data integration and management.

dbt (Data Build Tool): Transform data inside the warehouse with SQL-based workflows.

### ***7. Cloud and Resource Management***

Kubernetes: Container orchestration for managing scalable deployments.

Terraform: Infrastructure as code (IaC) tool for provisioning scalable infrastructure.

Cloud Dataflow: Google's managed service for stream and batch processing pipelines

## Development of Scalable Applications for Big Data Processing:

### REAL CASE STUDIES:

- Netflix: Real-time pipelines with Spark and Kafka for recommendations.
- Uber: Scalable architecture for ride-matching and analytics.
- Twitter: Handling millions of tweets/second with distributed systems.

## Development of Scalable Applications for Big Data Processing:

### STEPS TO BUILD BIG DATA APPLICATIONS:

1. Design modular data pipelines (ingestion, processing, storage).
2. Test with real-world data to identify bottlenecks.
3. Build fault-tolerant systems with recovery mechanisms.
4. Implement data partitioning for distributed workloads.

### CONCLUSION

- Scalability ensures big data systems grow with business needs.
- Proper design supports real-time analytics and batch processing.

## Development of Scalable Applications for Big Data Processing:

### QUESTIONS OR THOUGHTS?

- Discuss ideas on scalable big data applications.
- Explore strategies for your projects.

## 4. Types of Big Data processing to model business logic

---

## Real use case:

You are data analysts working for a transportation company, assisting area managers with reporting tasks.

The fleet manager reaches out to you, explaining that they need to annually submit information on the total kilometers traveled by each truck to a government agency.

The company has vehicles registered under two separate entities:

Company A and Company B.

Unfortunately, the telemetry data for the vehicles was lost during a database migration. You come up with the idea of retrieving this information from the daily CSV reports that are emailed.

You manage to extract these files from the emails and save them in the directory:

ETL-Python/data

Teams of 4.

### Phase I:

1. Analyze the downloaded data and determine which files contain the information needed to meet the fleet manager's requirements.
2. Create a single CSV file combining all relevant files (using Python) and save it in your sandbox at the following location: ETL-Python/data/Output/typefilename\_combined.csv
  - i. Each team will work in 2 pairs: 2 people uses python and 2 PowerQuery (Excel).

Analyze the combined content and provide a summary of the information.

### Phase II:

Design the Data Architecture to ensure the company collects the information needed to their use

## Real use case solution:

### Find the key data points:

As you were able to see, we need at least this two informations

- Distance
- Odometer

**Why?** Sometimes there are jumps in the odometer, and needs to be tackled with the addition of Distance. The Odometer max - min from the year has to be very close to the Sum of all distances registered.

**Design the Data Pipeline** to ensure you have always the most suitable. Choose the appropriate tools to guarantee a good job.

1. Collect all data
2. Process it
3. Load it into a data base or a semistructured document
4. Present it

# 05. Models, architectures, tools, and high-level languages for massive data processing.

# Cluster Computing

## Types of Clusters

- High availability  
Minimize shutdown time and provide uninterrupted service when a node fails.
- Load Balancing  
They are designed to distribute the workload among nodes, ensuring that tasks are shared and executed as soon as possible. This way, if any node fails, the workload is balanced across other nodes to complete the task

# Cluster Computing

## Cluster Structure

- **Symmetric**

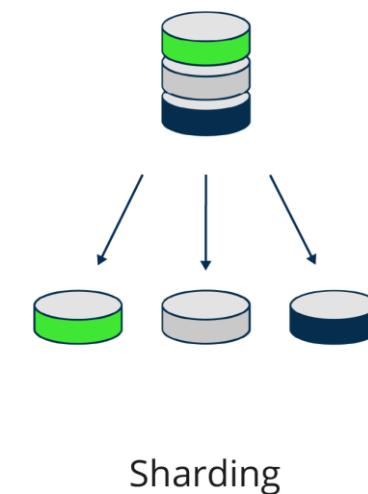
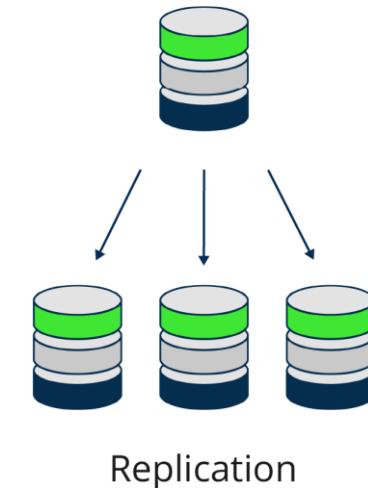
Symmetric clusters are those in which each node functions as an independent computer, capable of running applications. They are part of a subnet, and additional computers can be added without issues.

- **Asymmetric**

Asymmetric clusters have a head node that connects to all the worker nodes. In this configuration, the system depends on the head node, which acts as a gateway to the data/worker nodes.

## Distribution Models

- Replication
  - A copy of the same dataset is stored across different nodes.
- Sharding
  - The process of partitioning and distributing data across different nodes. Two partitions of the same data are never placed on the same node.
- Replication & Sharding
  - Both can be used independently or together.



# Distribution Models

## Replication Models

### Master-Slave Model

In this model, a master node controls the replication process, sending copies of the data to slave nodes and keeping track of where each replica is stored. If the master node fails, the entire system may become inoperable. To mitigate this risk, a secondary master node (failover master) should be implemented to take over in case of failure, ensuring continuous operation. This model is commonly used for read-heavy workloads, where the master handles updates, and slaves serve read requests.

### Peer-to-Peer Model

In this model, there is no central master node. Instead, all nodes are equal, and replication is typically used for read operations rather than writes. This approach enhances redundancy and load balancing, making it ideal for distributed systems where high availability is a priority. However, without a central authority, consistency management can be more complex, requiring conflict resolution mechanisms.

## Key Considerations

- **Performance & Scalability:** The Master-Slave model is efficient for structured workloads but introduces a single point of failure, whereas Peer-to-Peer systems provide more resilience at the cost of complexity.
- **Use Cases:** Master-Slave is often used in traditional databases and enterprise applications, while Peer-to-Peer is more common in decentralized networks, content distribution systems, and blockchain-based architectures.

# Distributed systems

## Examples:

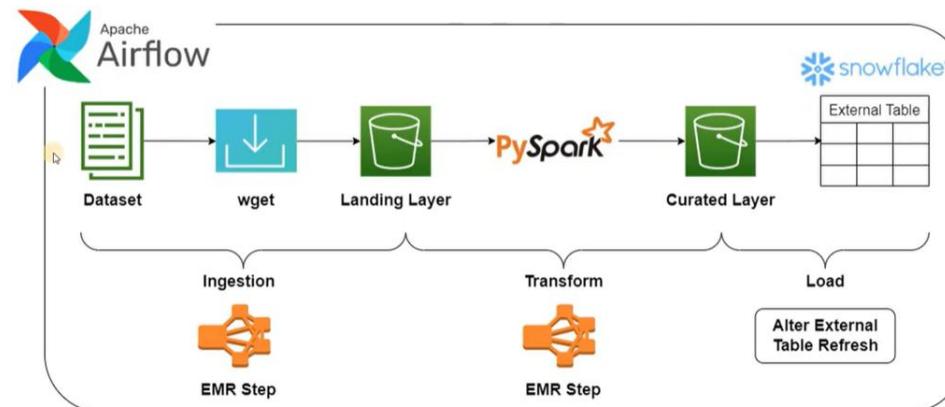
### Storage:

Explanation of HDFS and how Distributed Systems work: [link](#)

### Processing:

#### ETL Pipeline:

AirFlow, Spark [EMR (cloud computing)] + Load in Snowflake [Data Warehouse solution (cloud storage)] [link](#)



# When Distributed Systems May Not Be the Right Solution

## **Transactional Workloads with Random Data:**

If the task involves processing jobs in a transactional manner with unpredictable data access patterns, distributed systems may introduce unnecessary complexity and overhead.

**Non-Parallelizable Workloads:** When tasks cannot be broken down and executed in parallel, distributing them across multiple nodes does not provide any advantage and may even degrade performance.

**Low-Latency Data Access Requirements:** If the system demands extremely fast access to data with minimal delays, a centralized architecture or in-memory processing might be a better choice than a distributed approach.

**Handling a Large Number of Small Files:** Distributed systems are optimized for large-scale data processing, but managing a high volume of small files can introduce inefficiencies due to metadata overhead and excessive disk I/O operations.

**Intensive Computation with Minimal Data:** When workloads involve heavy computations but operate on small datasets, the cost of data transfer and synchronization across nodes can outweigh the benefits of distribution, making a local or specialized computing solution more efficient.

## Use Case: SmartHome & Buildings - IOT world

Intro to SmartHome & Building system.

What data we can get

What we do with it?

Let's think!!

# High level programming languages

## • Python

- Widely used due to its ease of use and extensive ecosystem of libraries for data processing (e.g., Pandas, Dask, PySpark).
- Supports machine learning and data analytics frameworks like TensorFlow and Scikit-learn.

## • Scala

- Designed for functional and object-oriented programming, making it ideal for distributed data processing.
- Native language for **Apache Spark**, ensuring high performance and efficient parallelism.

## • Java

- Strongly typed and widely adopted for enterprise-level applications.
- Used in Hadoop and Apache Beam, providing scalability and robustness.

## • R

- Preferred in statistical computing and data visualization.
- Integrates well with **SparkR** for large-scale analytics.

## • SQL (Structured Query Language)

- Essential for querying large datasets in **distributed databases** like Hive, Presto, and Google BigQuery.
- Used for structured data transformations and aggregations in big data pipelines.

## • Julia

- High-performance computing language optimized for numerical analysis.
- Growing adoption in big data analytics and machine learning.

## • Go (Golang)

- Designed for high concurrency and performance, making it useful in distributed computing.
- Used in large-scale data processing systems like **InfluxDB** and **CockroachDB**.

## • Rust

- Memory-safe and optimized for performance, making it suitable for large-scale real-time data processing.
- Used in distributed data platforms like Vector and Timely Dataflow.

# High level programming languages

**PERSONAL RECOMMENDATION:** Start with Python and SQL

## Where to Begin

1. Learn Python and SQL: Python is versatile, easy to learn, and widely used in data processing. SQL is essential for querying and managing large datasets efficiently.
2. Work on Practical Projects: Apply what you learn to real-world projects that interest you. Start with simple data analysis, ETL pipelines, or automation scripts.
3. Build a Structured Programming Mindset: Understanding how to structure your code and solve problems is more important than just learning a language. A well-organized thought process will help you adapt to new tools and technologies.
4. Expand Your Knowledge When Needed
  - If a project requires another language or tool, you'll be prepared to learn it quickly. AI-powered development tools will make learning and implementation even easier in the future.

## Final Thought: The Idea Matters Most

The most critical aspect is having a clear idea of what you want to achieve. Technology is just a tool—what really matters is how you use it to solve problems and create value.

GOOD LUCK AND ENJOY YOUR RIDE! 