

# Crypto Real-Time Streaming Dashboard

Progetto finale TAP 2025

Gabriele Privitera  
1000031845



# **Indice**

- Il Problema
- La Soluzione
- Architettura e Flusso
- Tecnologie Utilizzate
  - WebSocket API - Bybit
  - Apache Kafka
  - Apache Spark
  - Machine Learning
  - Elasticsearch
  - Kibana
  - Docker
- Demo Live
- Conclusioni e Possibilità di Espansione



# ***Il Problema***

Nel mondo delle criptovalute, i dati di mercato sono generati in quantità massicce e in tempo reale. Tuttavia, per analizzarli efficacemente e in modo tempestivo servono strumenti capaci di:

- raccogliere i dati da fonti esterne in streaming,
- processarli rapidamente,
- salvarli in un sistema scalabile e interrogabile,
- visualizzarli in dashboard intuitive.

Molti strumenti esistenti offrono queste funzioni separatamente, ma pochi offrono una pipeline completa e personalizzabile per progetti di analisi in tempo reale.





# La Soluzione

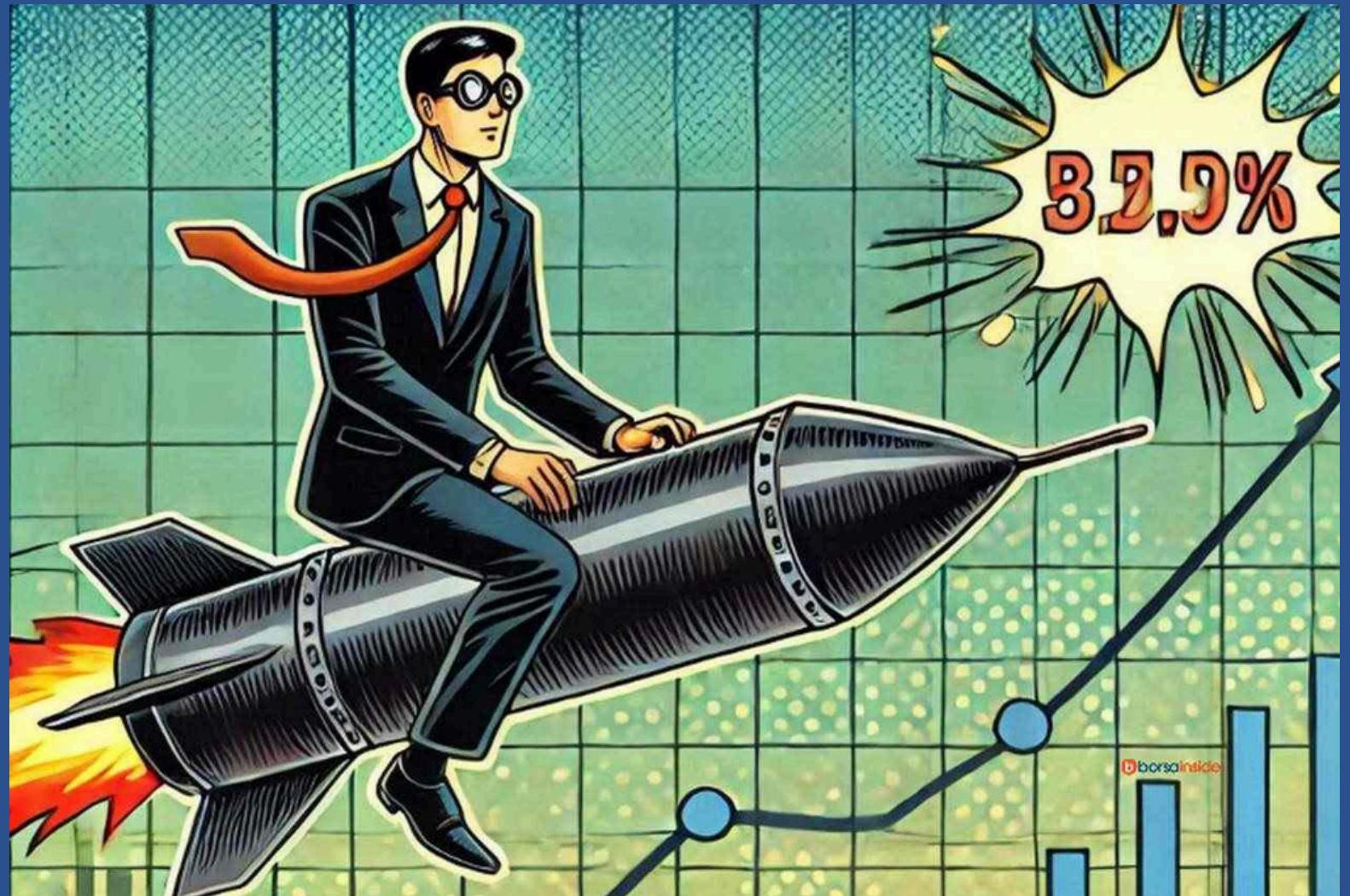
Realizzare una pipeline di streaming dati in tempo reale che raccolga informazioni sui trade pubblici da una o più piattaforme di criptovalute, li trasmetta attraverso un sistema di messaggistica, li elabori in tempo reale, li archivi in un motore di ricerca ottimizzato per grandi volumi di dati e ne consenta la visualizzazione tramite una dashboard interattiva. In questo modo:

- Puoi analizzare in tempo reale il comportamento di mercato
- Identificare pattern, volumi, anomalie
- Creare dashboard dinamiche per supportare decisioni data-driven

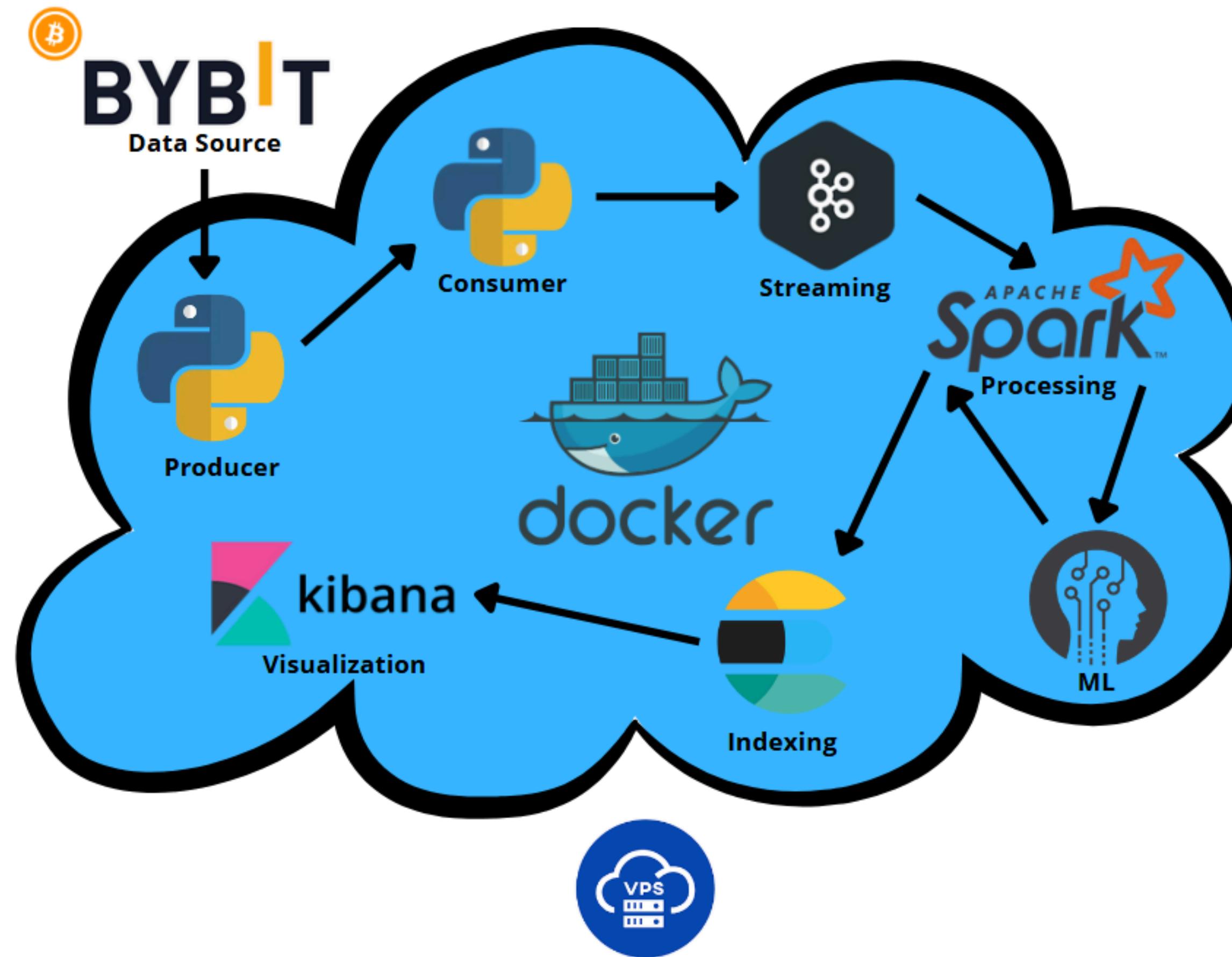
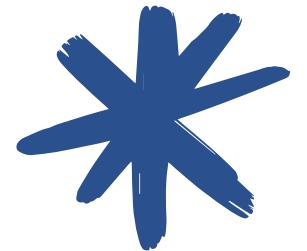


*Da questo nasce:*

# *Crypto Real-Time Streaming Dashboard*

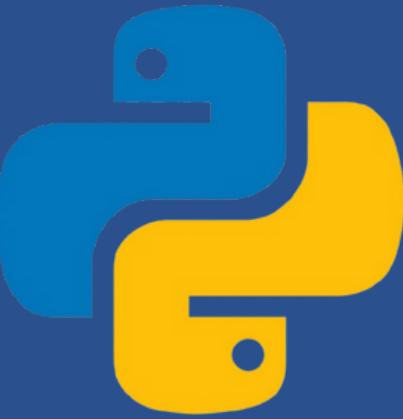


# Architettura e flusso dati



# *Tecnologie Utilizzate*

# *WebSocket API - Bybit*



Ruolo: Fonte dei dati in tempo reale

- Utilizza la Bybit WebSocket API per ricevere transazioni pubbliche istantanee sui mercati spot crypto.
- Ogni messaggio rappresenta un trade (acquisto/vendita) eseguito su una criptovaluta.
- È stata scelta per la bassa latenza, l'aggiornamento costante e l'ampia copertura di coppie crypto/USDT.
- Gestione avanzata tramite batch e thread per superare limiti di sottoscrizione.



# *Apache Kafka*

Ruolo: **Broker di messaggi / buffer intermedio**

- Kafka riceve i dati dalla WebSocket e li mantiene in un topic (bybit-trades).
- Garantisce resilienza, scalabilità e un flusso continuo verso Spark.
- È perfetto per scenari di streaming real-time, dove servono durabilità e ordine nei messaggi.
- Kafka è il cuore del flusso: disaccoppia la sorgente (Bybit) dal processamento (Spark).



# ***Apache Spark Streaming***

**Ruolo: Processamento dei dati in tempo reale**

- Consuma i messaggi dal topic Kafka e li elabora con micro-batch.
- Esegue trasformazioni, filtri, aggregazioni e arricchisce i dati con predizioni ML.
- Utilizza PySpark per scrivere facilmente job scalabili.
- Invia i dati puliti e predetti su Elasticsearch per l'analisi visiva.



# Machine Learning

## Ruolo: Predizione della direzione del prezzo

- È stato addestrato un modello ‘Random Forest’ tramite ‘scikit-learn’.
- Il dataset è stato creato custom esportando i dati reali da Elasticsearch, aggiungendo un campo target:
  - 1 se il prezzo aumenta dopo il trade, 0 altrimenti.
- Dopo l’addestramento, il modello è servito da un microservizio FastAPI.
- Ogni batch Spark invia i dati al servizio /predict e riceve la predizione.
- Le predizioni vengono salvate insieme ai dati su Elasticsearch per analisi avanzate.



# *Elasticsearch*

## Ruolo: Archiviazione e indicizzazione dei dati

- Database NoSQL ottimizzato per la ricerca veloce e analisi su grandi volumi.
- I dati sono indicizzati per campo, rendendo semplici e rapide le interrogazioni.
- Supporta aggregazioni temporali, testuali, numeriche.
- Perfetto per integrare con Kibana in tempo reale.



# *Kibana*

## Ruolo: Visualizzazione dei dati

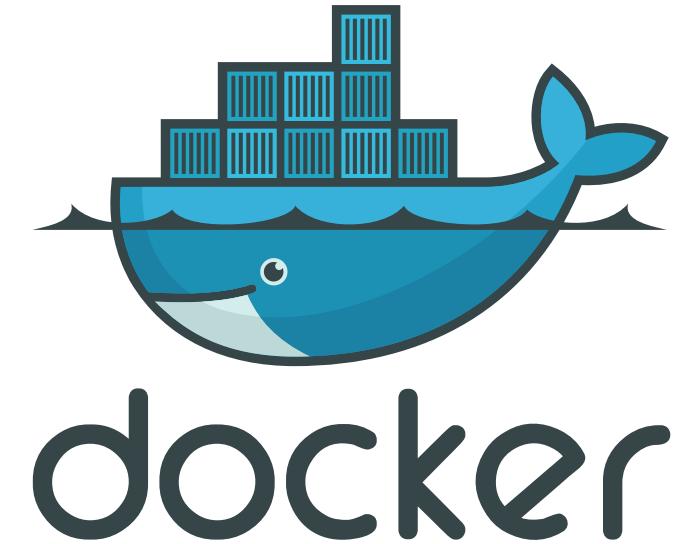
- Interfaccia grafica per creare dashboard dinamiche e personalizzate.
- Usata per visualizzare in tempo reale:
  - Prezzi, volumi, direzione dei trade, predizioni del modello, distribuzioni.
- Ogni visualizzazione si basa sui dati di Elasticsearch.
- Permette drilldown interattivi e filtri live su simbolo, tempo, prediction, etc.



kibana

# **Docker**

## Ruolo: Isolamento e portabilità dei servizi



- Ogni componente del progetto (Kafka, Spark, Elasticsearch, FastAPI, ecc.) gira in un container Docker.
- Questo garantisce:
  - Portabilità del progetto su qualsiasi sistema operativo
  - Semplicità nel deploy e nella gestione dei servizi
  - Isolamento tra le varie tecnologie
- È stato utilizzato anche per l'addestramento ML e la gestione del flusso completo.
- Attualmente il progetto è deployato su una VPS privata, permettendo test e accesso da remoto in ambiente di produzione simulato.



E adesso il momento più temuto di tutta la presentazione:

# ***DEMO LIVE***



# *Conclusioni e possibilità di espansione*

Il progetto mostra come si può creare una pipeline di streaming scalabile e flessibile, utile per:

- Il **monitoraggio** in tempo reale del mercato crypto
- L'**analisi** dei trend e delle fluttuazioni
- Lo **sviluppo di modelli predittivi** per supporto decisionale

**Possibili estensioni future:**

- **Addestramento continuo** del modello (online learning)
- Integrazione con Telegram per **alert automatici**
- **Analisi storiche più avanzate** (ex: rolling windows)
- Integrazione con **ulteriori indicatori tecnici** (RSI, ...)



**Graxie per  
l'attenzione!**

