# Optimal Approaches for Real-Time Machine Learning with Apache Spark on Kubernetes: Best Practices and Strategies

Hichem Kenniche
Open-Source Software Product Architect

Now part of Spot by NetApp

# A few words about me!

Hichem Kenniche,

OSS Product Architect @Instaclustr (part of NetApp)

Previously at:

- Databricks
- Capgemini Invent
- Sony PlayStation

# Disclaimer

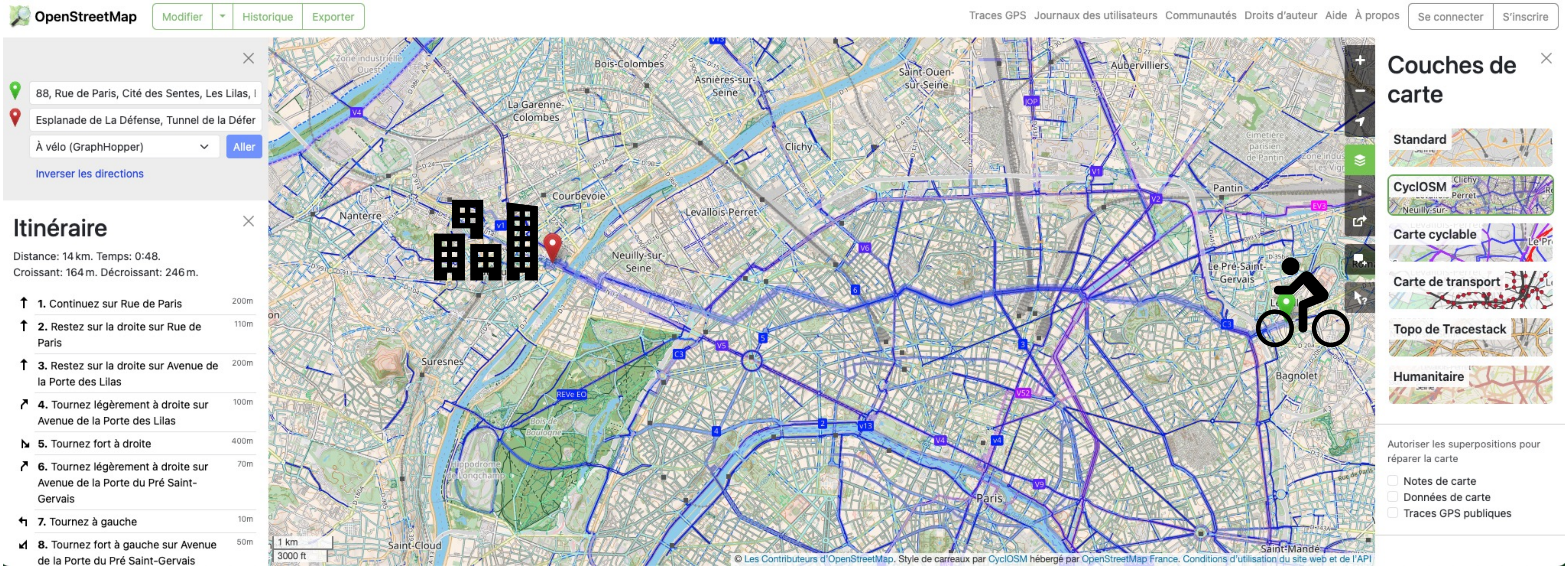
- This is not a contribution to any OSS project !
- My vision of things is necessarily biased !
- Most of this is work is based on the principles of OSS, open data, and a culture of knowledge sharing ♥
- (Human) Learning is a Lifelong WIP …

# Our agenda for today

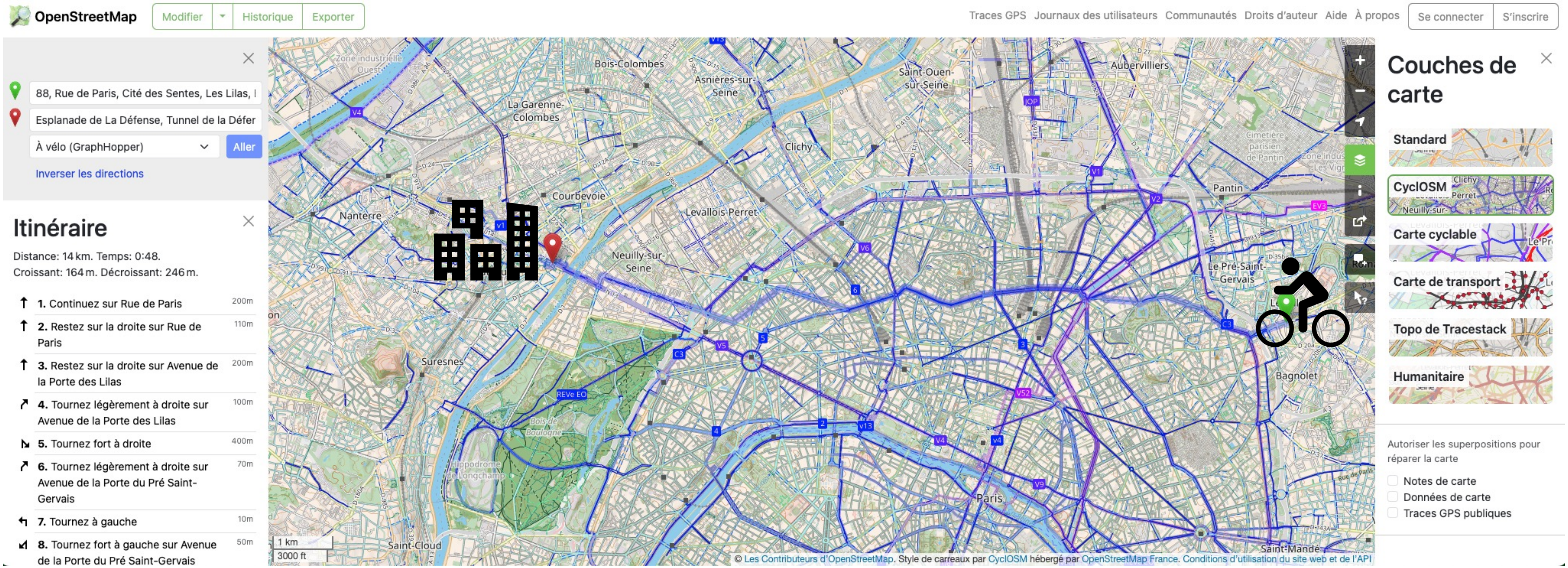
- The fundamentals of Real-Time ML.
- The biggest challenges facing Data teams.
- The motivation behind running Spark on Kubernetes
- Some challenges of Running Spark on Kubernetes and solutions
- Conclusion and takeaways

# The fundamentals of Real-Time ML.

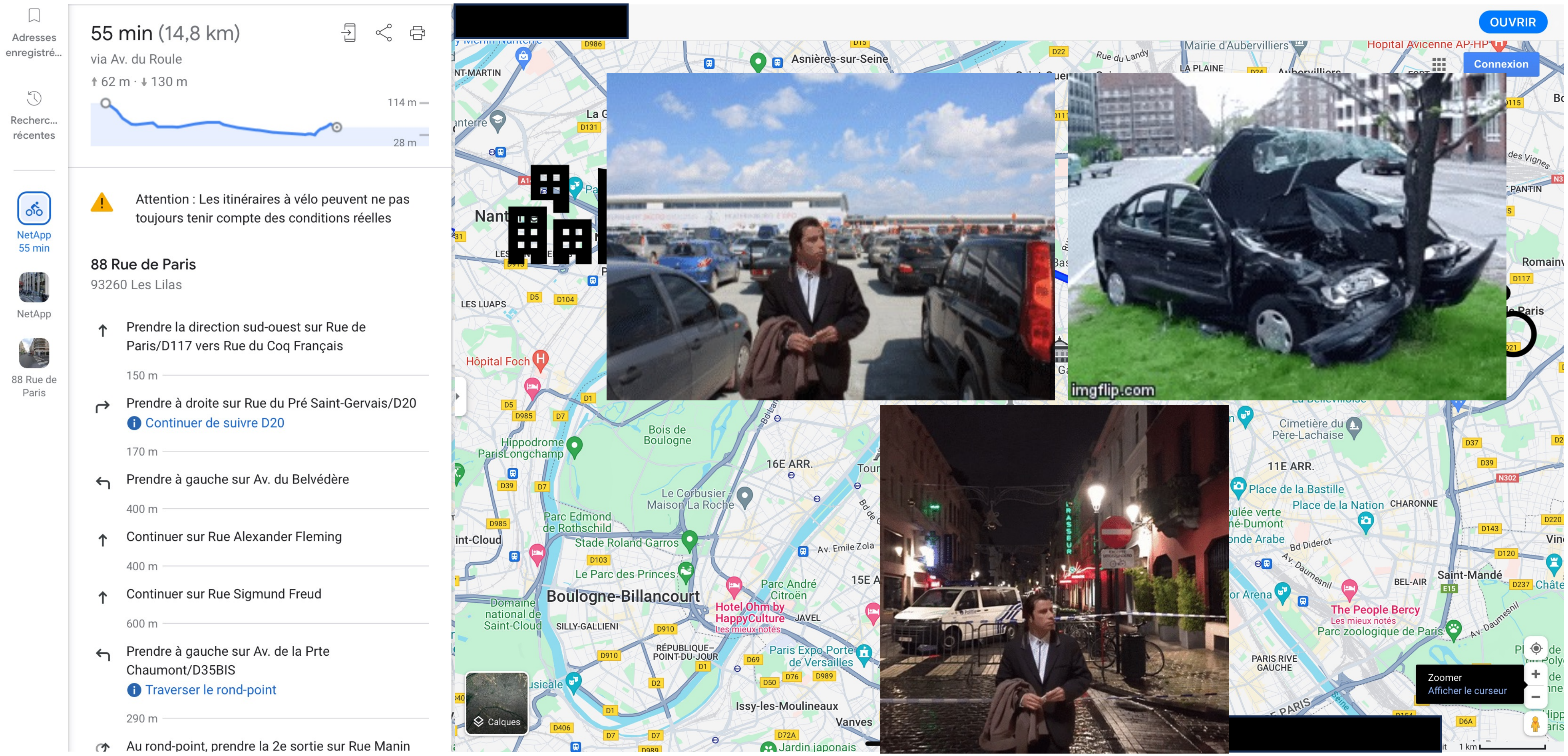
# Finding the shortest, fastest Cycling Route



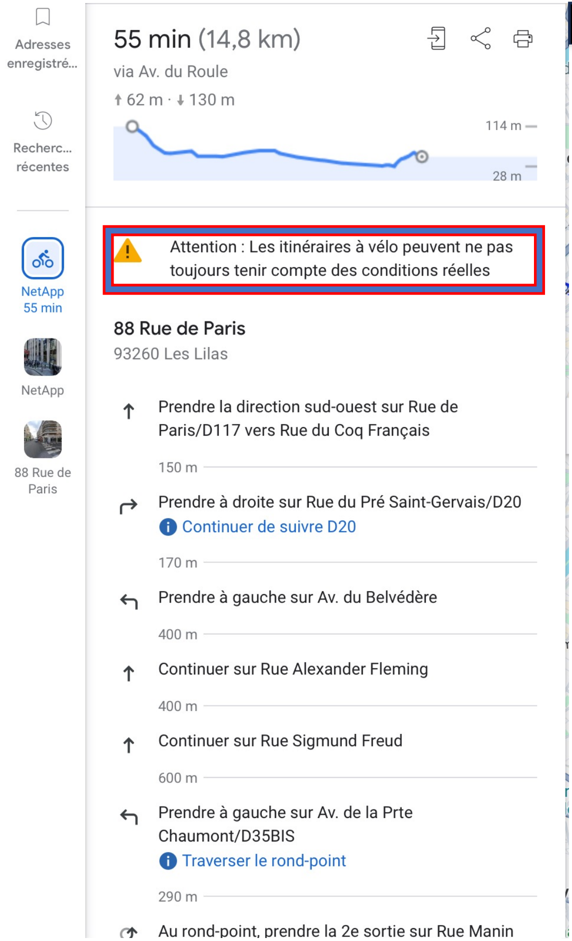
# Finding the shortest, fastest, least traffic?



# Finding the shortest, fastest, least traffic!



# Finding the shortest, fastest, least traffic!




[redacted]

Johanne Original Poster Aug 14, 2020

## How does [redacted] calculate the shortest / preferred cycling route in the Netherlands?

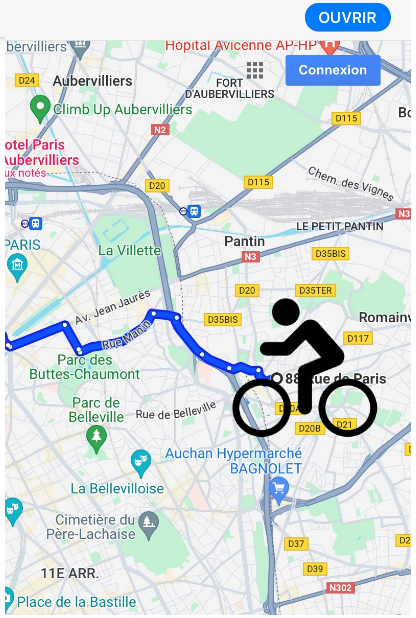
I live in the Netherlands and I often use [redacted] it to look up where I'm going and what route I need to take. It almost never gives me the shortest or best route; [redacted] is -at least in the Netherlands- extremely bad at showing cycling routes. I wonder why? Calculating the shortest route by taking all available bike paths/roads and getting the shortest within the network should be a very easy thing to do.

Often when having been shown a route, I will then proceed to -while on my computer- drag it around a bit and often decrease the distance by about 10% of the total distance. While doing this I also see that it actually recognises these roads as cycling paths, but just doesn't use or suggest them.

I also notice that it there are many roads s
NEVER suggested by [redacted]. So I'm
roads, since they are actually the most su
been there for many years to have been n

I'm just really curious how [redacted] ge
that, how I can help to improve them? Per
behind, or maybe it has to do with the unc

Details
Directions and Navigation, Desktop - Other

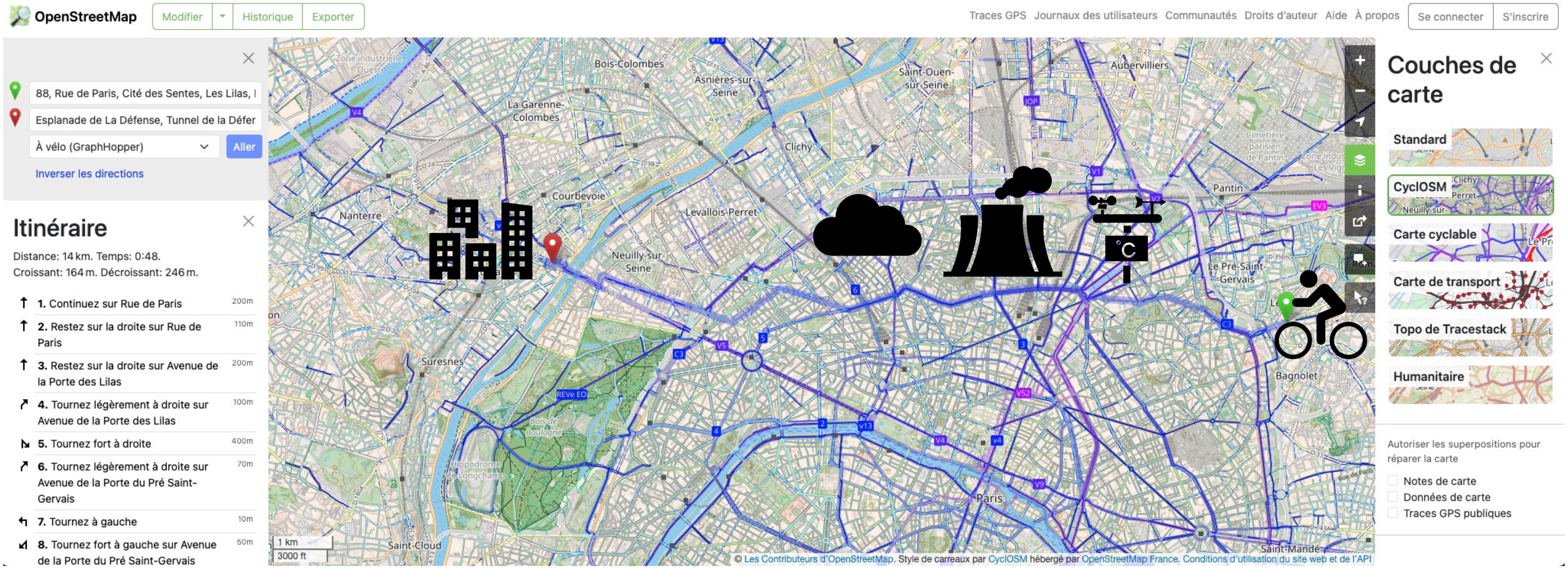



All Replies

Saket Pofali Sep 4, 2020

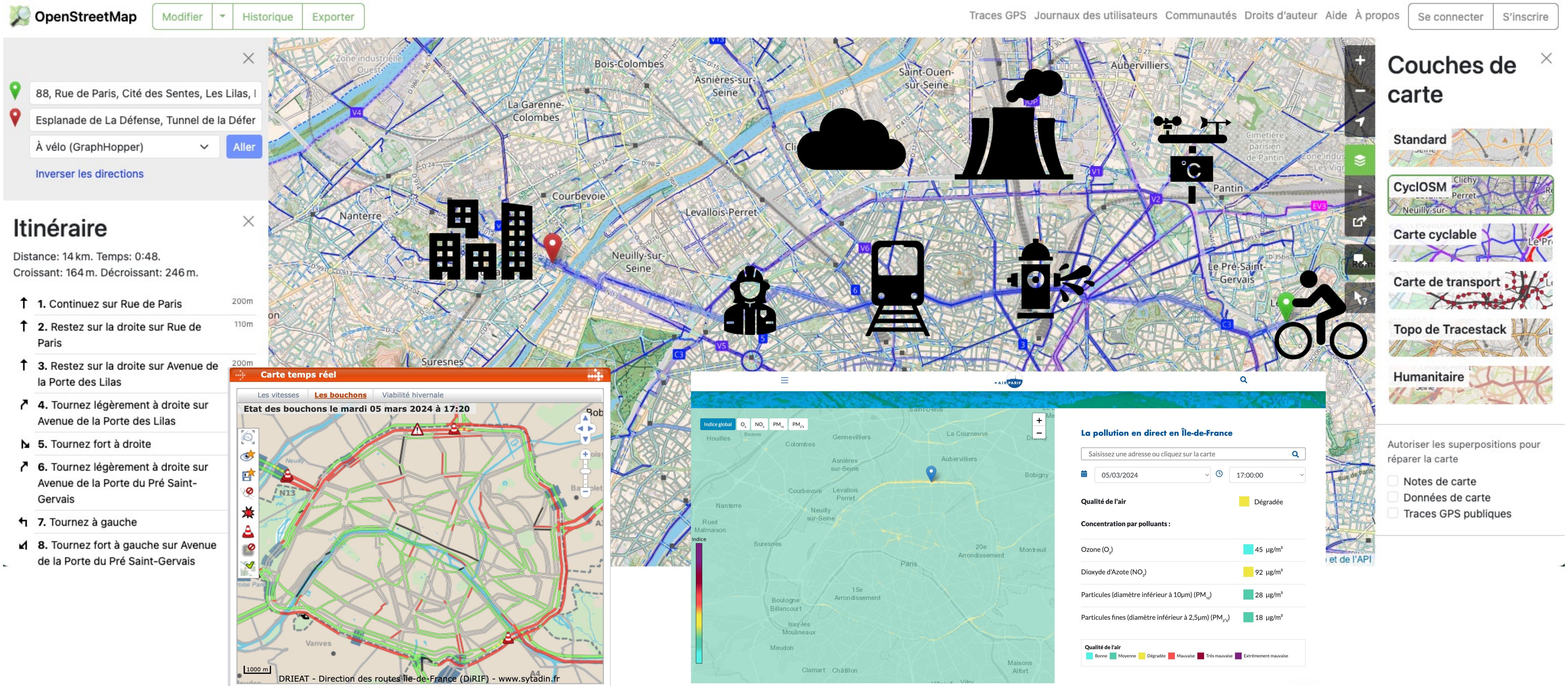
I've noticed this as well. I think [redacted] algorithms still based on recognizing roads and streets meant for motorized vehicular traffic which would mean that there would be minimum road width would be required to recognized as legitimate street.

# Finding the Least Air Pollution Exposure Cycling Routes



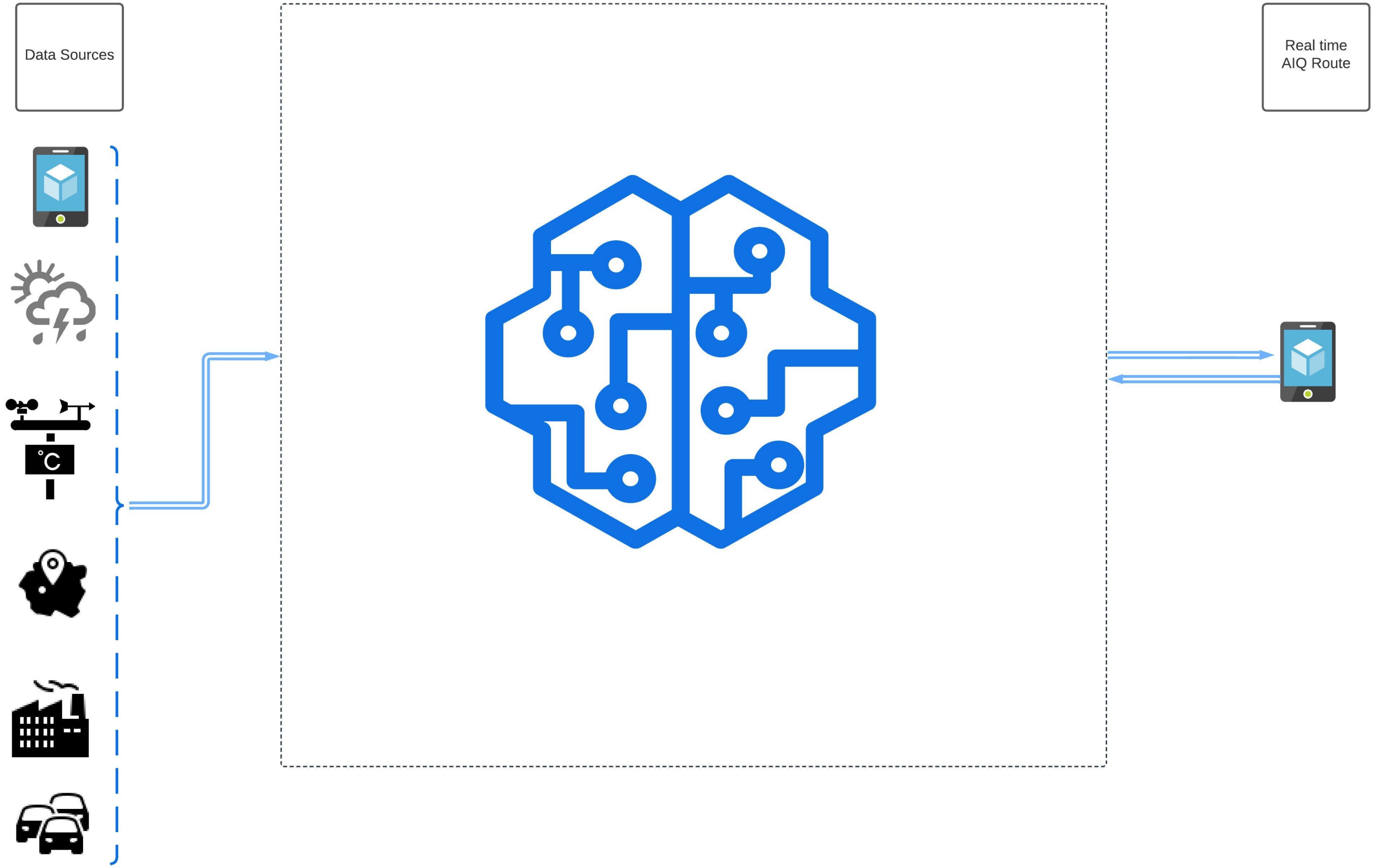
# Finding the Least Air Pollution Exposure Cycling Routes

# Finding the Least Air Pollution Exposure Routes (in real time)

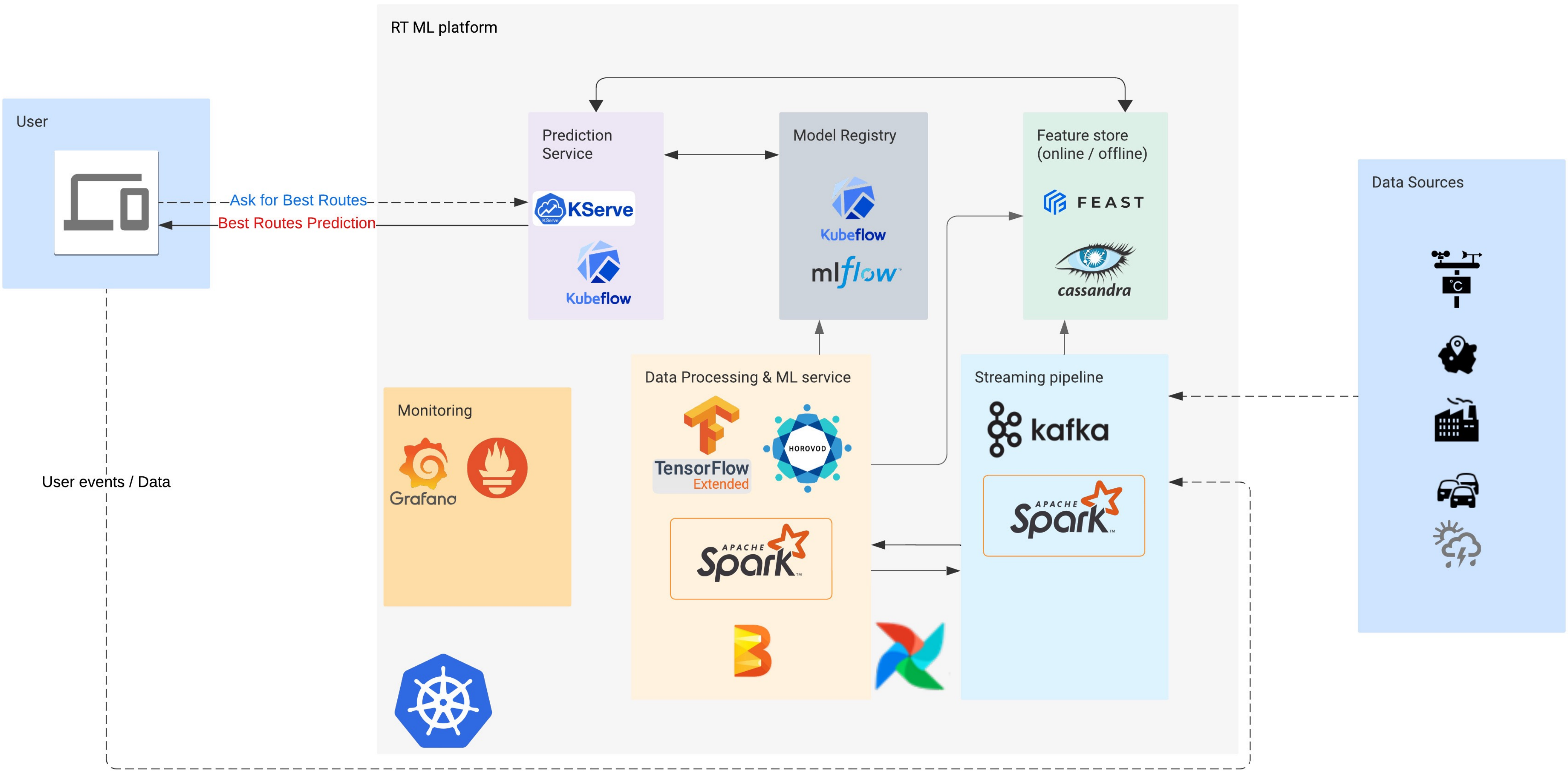


# Finding the Least Air Pollution Exposure Routes (in real time)

Real-time machine learning: the application of machine learning models to generate predictions or decisions in real-time and adapt to the changing environment.



# Real-time Machine Learning Platform



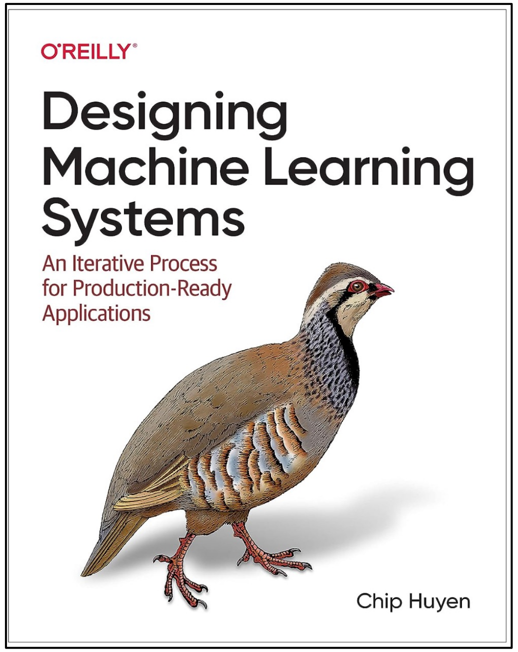
# The biggest challenges facing Data teams.

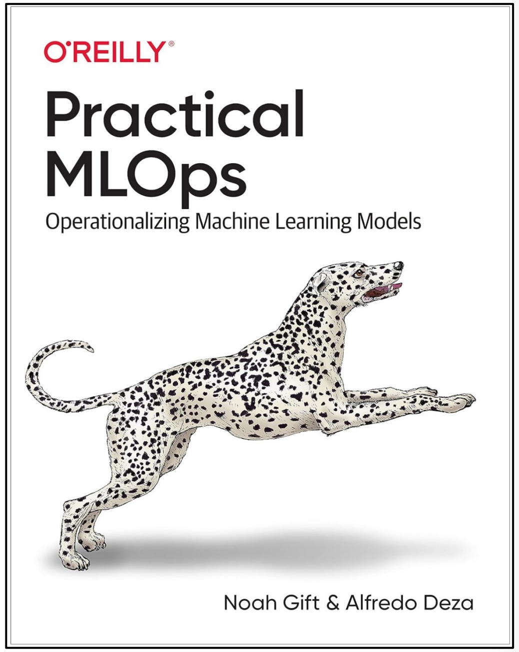
# Real-time Machine Learning Challenges


Chip Huyen    Blog    Books    List 100    MLOps Guide    Tiếng Việt

Real-time machine learning: challenges and solutions

Jan 2, 2022 • Chip Huyen




| | |
|---|---|
| Feature Engineering | Stream Processing |
| Incremental Learning (online learning) | Scalability |
| Model Updating | Latency |
| Model / Data Drift | Monitoring |
| Performance Evaluation | Distributed Training & Inference |
| MLOps | Resource Management / Cost |

# Real-time Machine Learning Challenges

Real-time machine learning challenges (our experience) are largely an infrastructure problem.

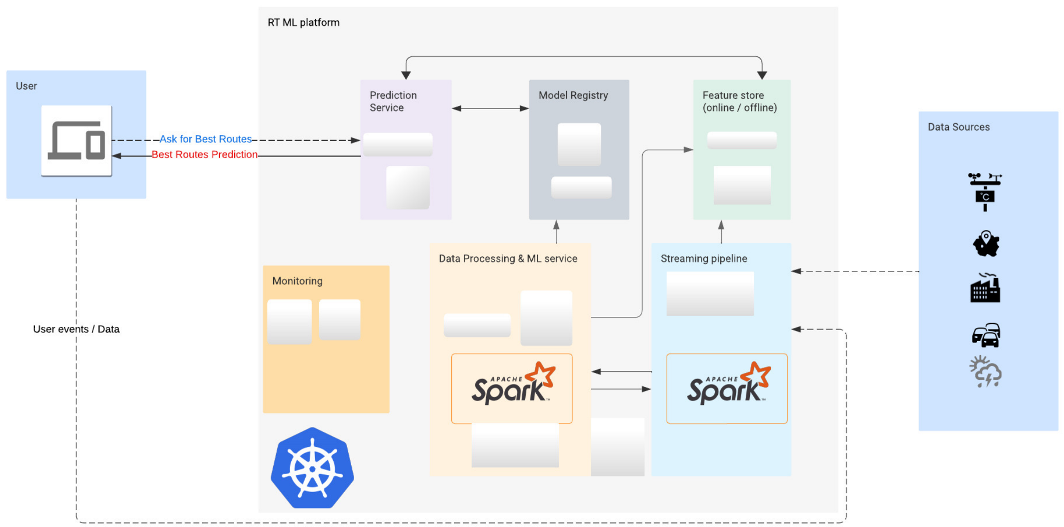
- Stream Processing
- Scalability
- Latency
- Monitoring
- Distributed Training & Inference
- Resource Management / Cost

# Solving some Real-time Machine Learning Challenges

Addressing these challenges requires a significant investment in advanced (OSS) technologies.

Spark on k8s:

- Stream processing
- Training
- Scalability & Latency
- Resource Efficiency



# The motivation behind running Spark on kubernetes

# Apache Spark is the #1 analytics engine for Big Data & AI

**Fast***
Massively parallelizable, efficient read and write

- Object stores
- Data warehouses
- Streams
- SQL/NoSQL databases

**Easy**
Interfaces with well-known programming languages

APACHE Spark

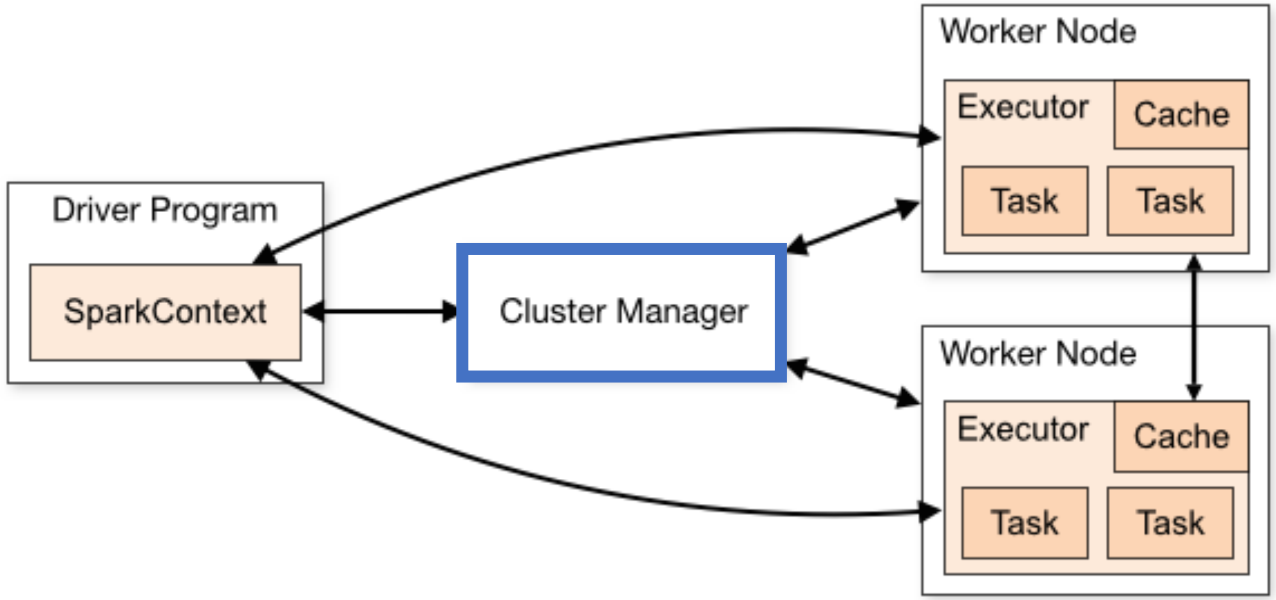
- Python
- Scala/Java
- SQL

**Versatile**
Across multiple use cases

- ETL/ELT
- Real-time
- ML
- BI

# The role of resource manager in a Spark cluster

Spark depends on cluster manager for orchestration of a job on a cluster



# Kubernetes is the latest cluster manager for Spark

- Standalone: built-in, limited functionalities
- Apache Mesos: deprecated as of Spark 3.2.0
- Hadoop YARN: most widely used today
- **Kubernetes: most popular among new deployments**

# The Spark on Kubernetes Journey

**Feb 2018 - Spark 2.3**
Initial support released for Spark on Kubernetes

**Nov 2018 - Spark 2.4**
Client Mode, Volume Mounts, PySpark and R support

**June 2020 - Spark 3.0**
Dynamic Allocation, Local code upload, Kerberos Support

**March 2021 - Spark 3.1**
Spark on Kubernetes generally available
Graceful node shutdown, NFS mounts, Dynamic Persistent Volume Claims

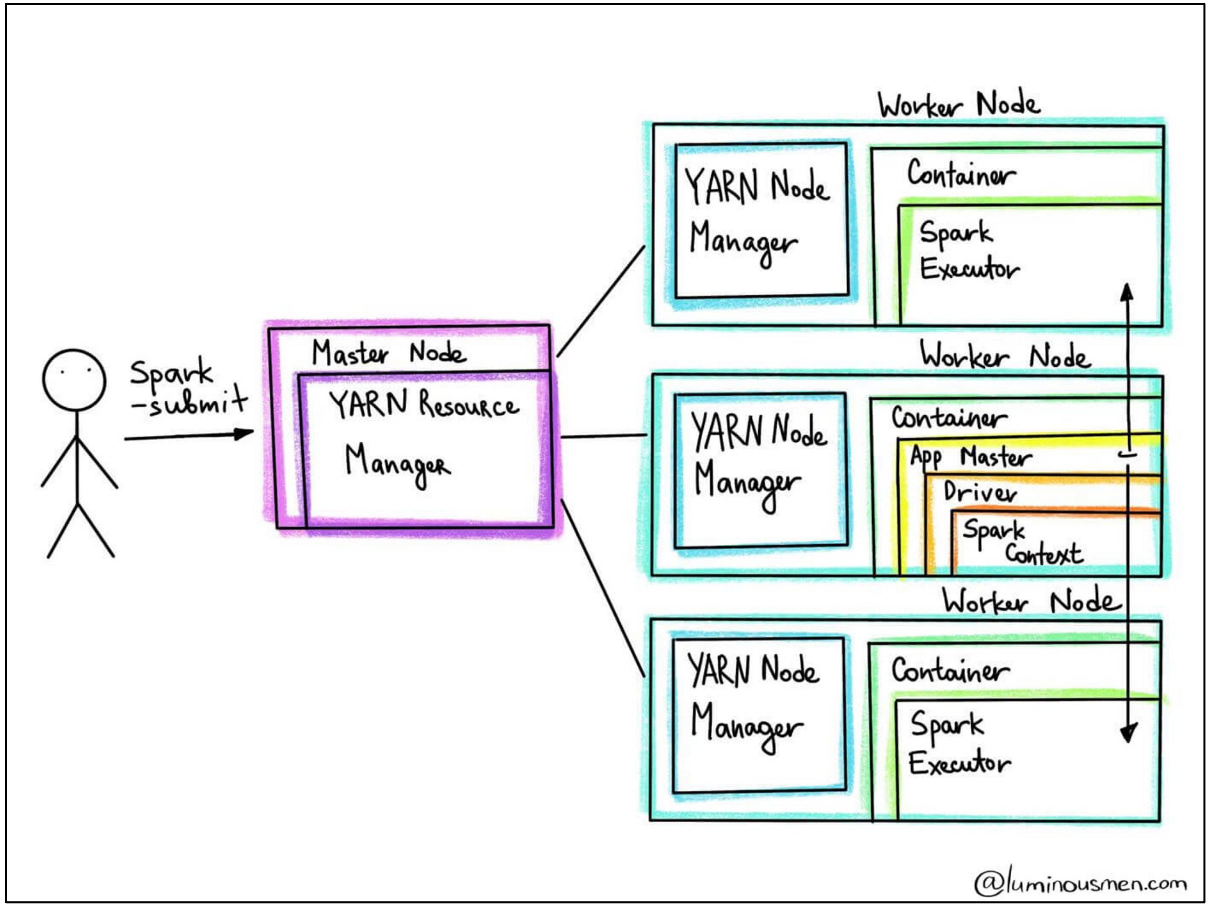
**Oct 2021 - Spark 3.2**
Dynamic PVC mounting and reuse, Faster S3 Writes (Magic Committer enabled)

**June 2020- Spark 3.3**
Executor Rolling in Kubernetes environment, Support Customized Kubernetes Schedulers

**Apr 2023 - Spark 3.4**
PVC-oriented executor pod allocation

**Spark 3.5**
Upgrade kubernetes-client

# Spark on YARN: architecture & pain points




**Global Spark version and shared libraries**

- You'll have a Spark 2.4 cluster, a Spark 3.0 cluster, a Spark 3.1 cluster.
- Transient clusters are recommended for stability, but increase costs.

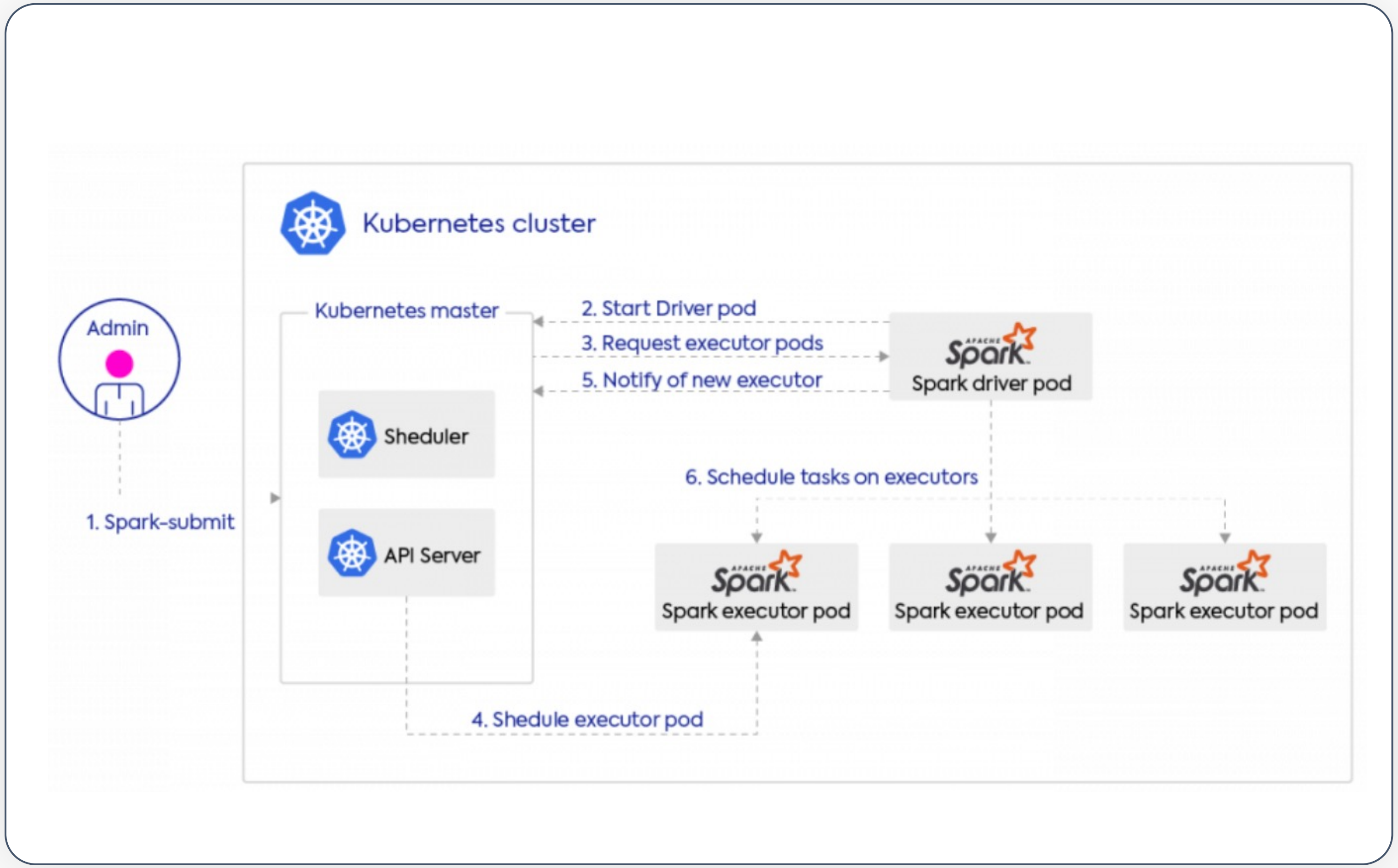
**Limited Docker image support***

- Environment is built from AMIs and bash scripts, flaky runtime library installation
- Debugging is painful - there's no way to run Spark locally, environment is subtle

**Resource Overhead**

- Slow startup time
- YARN master node, YARN Node Mgr are JVM processes using a lot of resources.

# Spark on Kubernetes: architecture & benefits




## Native Dockerization

- Simpler dependency management
- Reliable executions across environments (locally during development, staging, production)
- Faster startup time

## A single long-running cluster

- Quick to scale up (and down) based on load
- Mix different Spark versions
- Mix Spark and non-Spark apps
- Mix use cases (notebooks, batch/streaming jobs)

## A standard, agnostic infrastructure layer

- Reduce lock in
- Simplify your operations
- Leverage the open-source tools from the cloud-native ecosystem

# Two ways to run Spark apps on k8s

## Spark-submit

- “Vanilla” way from Spark main open source repo
- Requires Spark distribution on client
- Configs spread between Spark config (mostly) and k8s manifests
- Less pod customization support (improving)
- App management is more manual

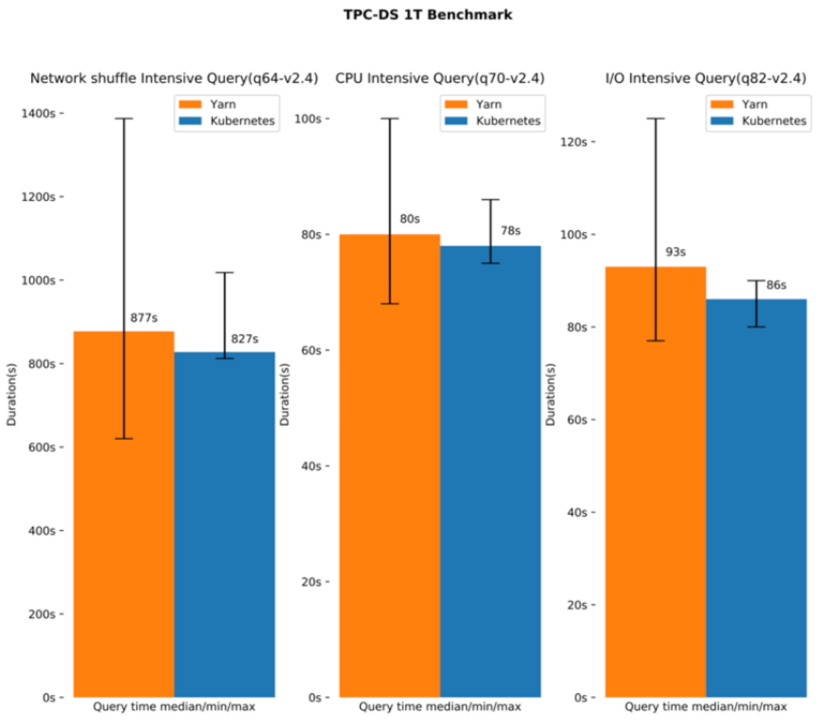
## spark-on-k8s operator

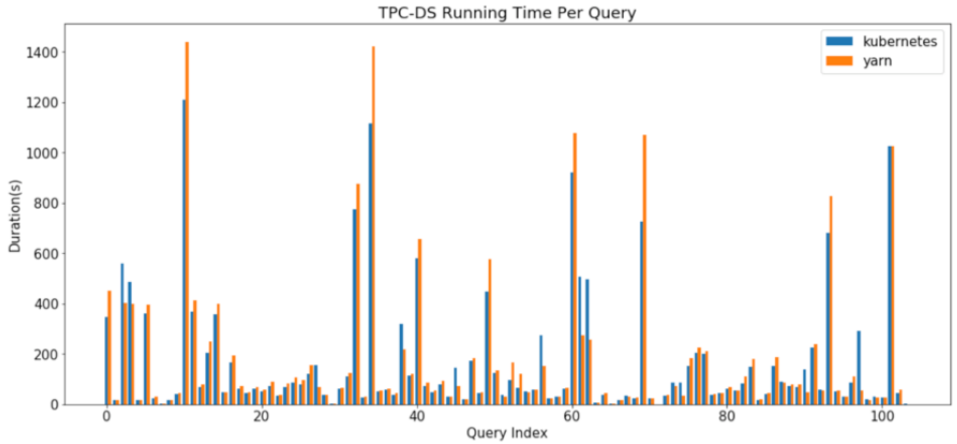
### Overview

The Kubernetes Operator for Apache Spark aims to make specifying and running Spark applications as easy and idiomatic as running other workloads on Kubernetes. It uses Kubernetes custom resources for specifying, running, and surfacing status of Spark applications. For a complete reference of the custom resource definitions, please refer to the API Definition. For details on its design, please refer to the design doc. It requires Spark 2.3 and above that supports Kubernetes as a native scheduler backend.

The Kubernetes Operator for Apache Spark currently supports the following list of features:

- Supports Spark 2.3 and up.
- Enables declarative application specification and management of applications through custom resources.
- Automatically runs `spark-submit` on behalf of users for each `SparkApplication` eligible for submission.
- Provides native cron support for running scheduled applications.
- Supports customization of Spark pods beyond what Spark natively is able to do through the mutating admission webhook, e.g., mounting ConfigMaps and volumes, and setting pod affinity/anti-affinity.
- Supports automatic application re-submission for updated `SparkApplication` objects with updated specification.
- Supports automatic application restart with a configurable restart policy.
- Supports automatic retries of failed submissions with optional linear back-off.
- Supports mounting local Hadoop configuration as a Kubernetes ConfigMap automatically via `sparkctl`.
- Supports automatically staging local application dependencies to Google Cloud Storage (GCS) via `sparkctl`.
- Supports collecting and exporting application-level metrics and driver/executor metrics to Prometheus.



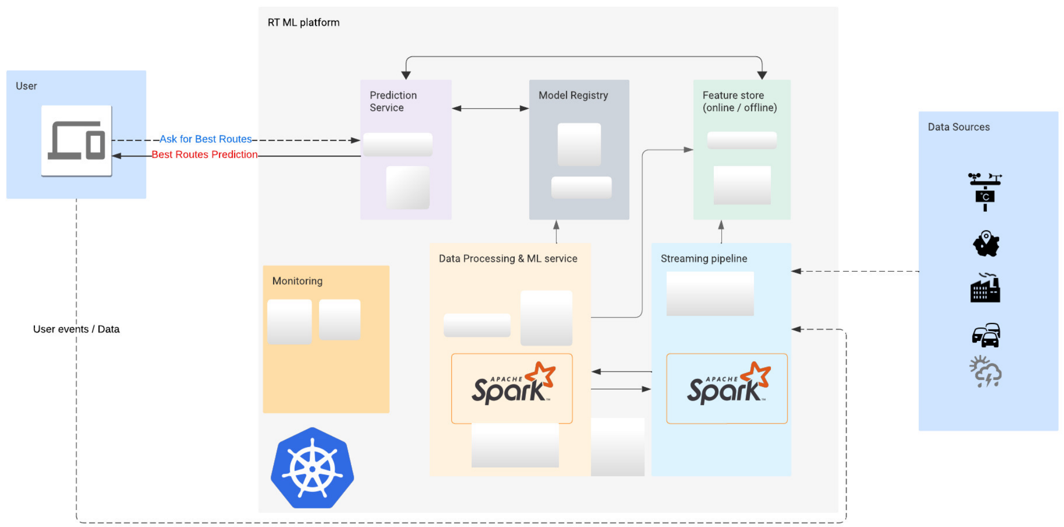




https://aws.amazon.com/fr/blogs/containers/optimizing-spark-performance-on-kubernetes/

# Some challenges of running Spark on kubernetes and solutions

# Challenges in the context of R-L M-L

- Monitoring
- Scalability
- Latency
- Models Training



# Monitoring: logs, logs and more logs

Key information is buried under a lot of noisy one.

- Spark event/driver/executor logs.
- kubernetes logs
- Hard to reconcile with Spark jobs/stages/tasks

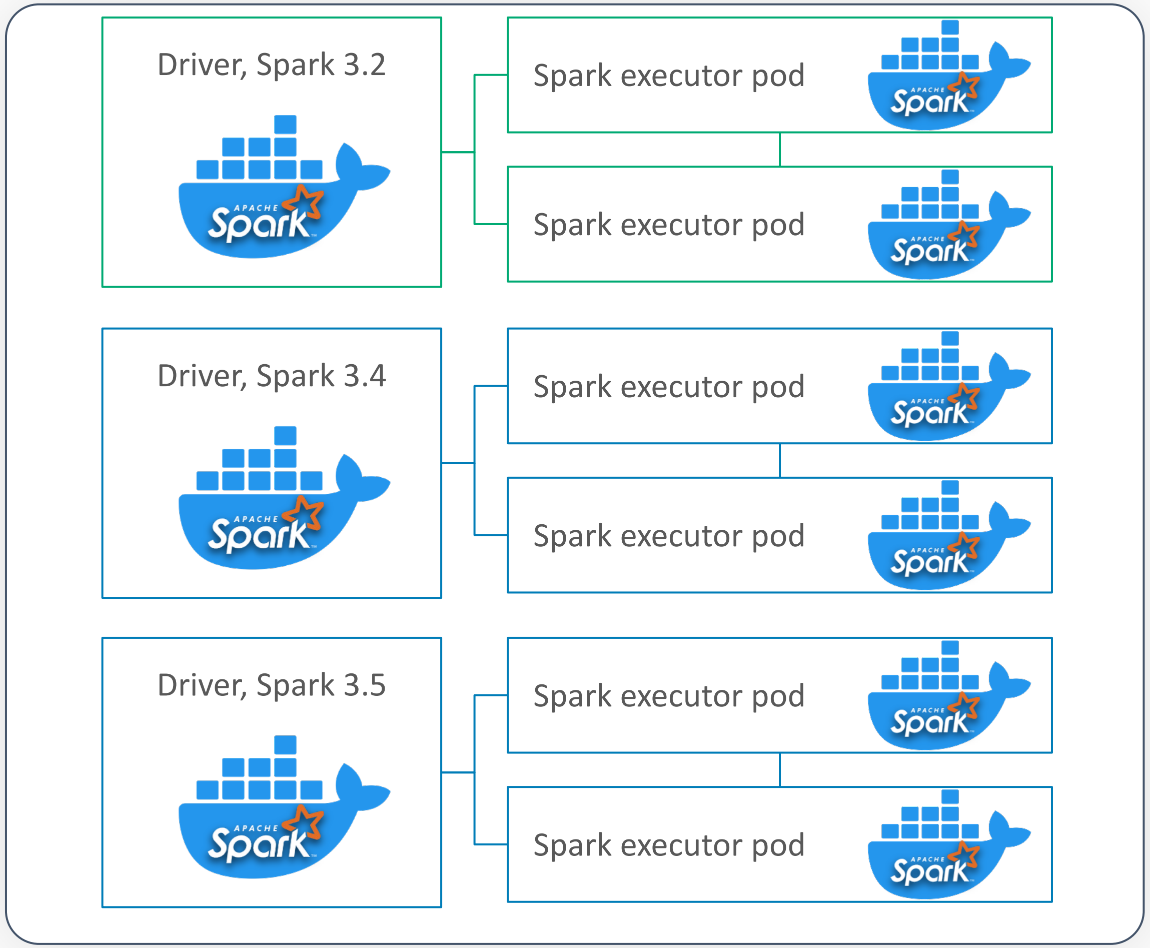
Solution

- Logs shipping tools : fluentbit & logstach
- Prometheus: Spark has a built-in Prometheus

# Scalability

Key factors to consider

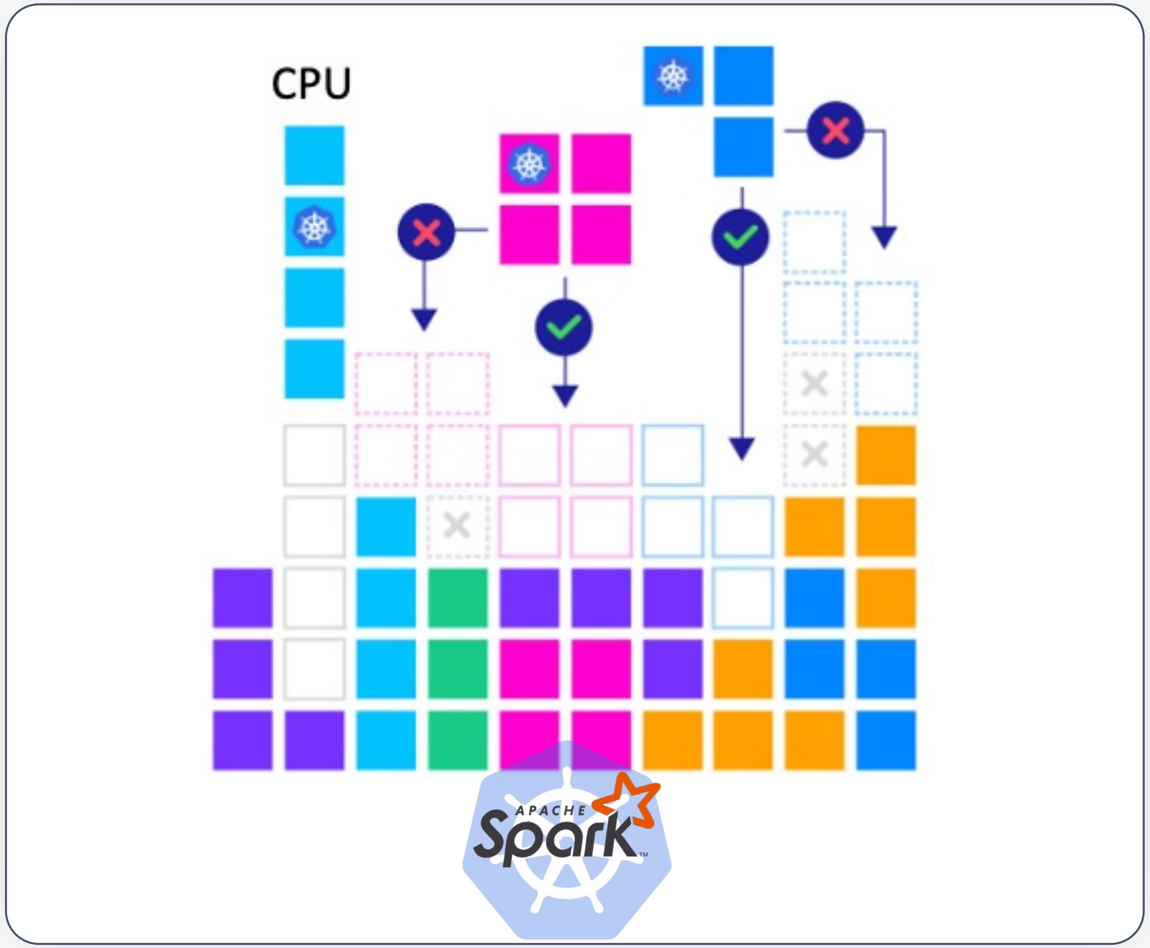
- Cluster sizing, infra choice/specs.
- Dynamic Allocation
- Shuffle data (NO external shuffle service YET )



# Scalability: the right sizing

For the sizing:

- Continuous and repeated exercise: know your data sources
- When selecting the cluster focus on enhancing parallelism in relation to the source.
- Streaming is CPU-bounded, State matters too (avoid spills)
- Deep Learning models with relatively long training and inference time: mix CPU with GPU (when required).



# Scalability: Dynamic Allocation ( A two-sided problem)

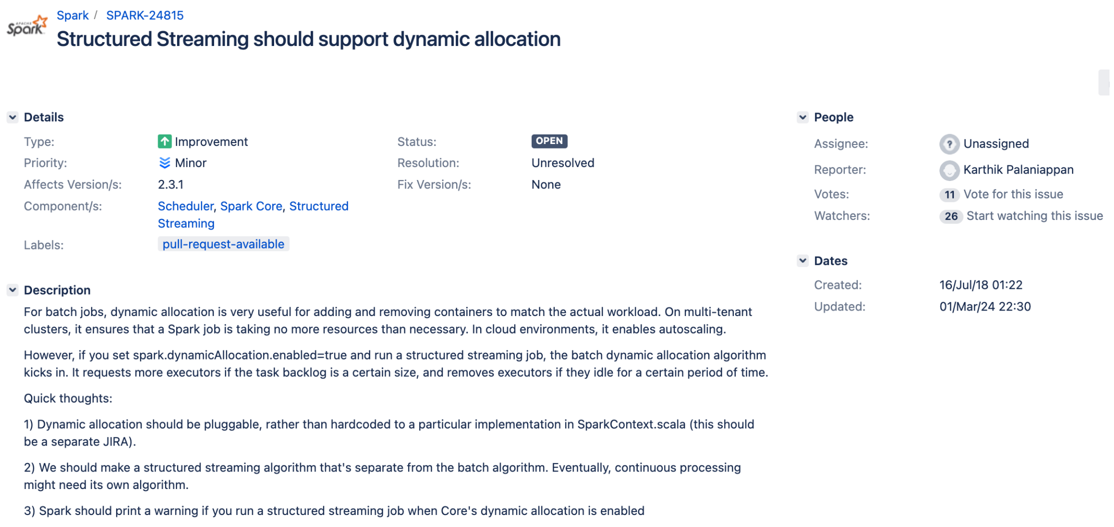
## Dynamic Allocation in Spark Structured Streaming

- Designed for batch jobs, it is compatible with batch and Spark structured streaming. Works poorly for certain applications !

## Dynamic Allocation (within k8s )

- This feature **may** cause issues with Spark Scalability on k8s !

[SPARK-24815 ] Structured Streaming should support dynamic allocation



Spark / SPARK-24815

Structured Streaming should support dynamic allocation

**Details**

| | | | |
|---|---|---|---|
| Type: | Improvement | Status: | OPEN |
| Priority: | Minor | Resolution: | Unresolved |
| Affects Version/s: | 2.3.1 | Fix Version/s: | None |
| Component/s: | Scheduler, Spark Core, Structured Streaming | | |
| Labels: | pull-request-available | | |

**People**

| | |
|---|---|
| Assignee: | Unassigned |
| Reporter: | Karthik Palaniappan |
| Votes: | 11 Vote for this issue |
| Watchers: | 26 Start watching this issue |

**Dates**

| | |
|---|---|
| Created: | 16/Jul/18 01:22 |
| Updated: | 01/Mar/24 22:30 |

**Description**

For batch jobs, dynamic allocation is very useful for adding and removing containers to match the actual workload. On multi-tenant clusters, it ensures that a Spark job is taking no more resources than necessary. In cloud environments, it enables autoscaling.

However, if you set spark.dynamicAllocation.enabled=true and run a structured streaming job, the batch dynamic allocation algorithm kicks in. It requests more executors if the task backlog is a certain size, and removes executors if they idle for a certain period of time.

Quick thoughts:

1) Dynamic allocation should be pluggable, rather than hardcoded to a particular implementation in SparkContext.scala (this should be a separate JIRA).

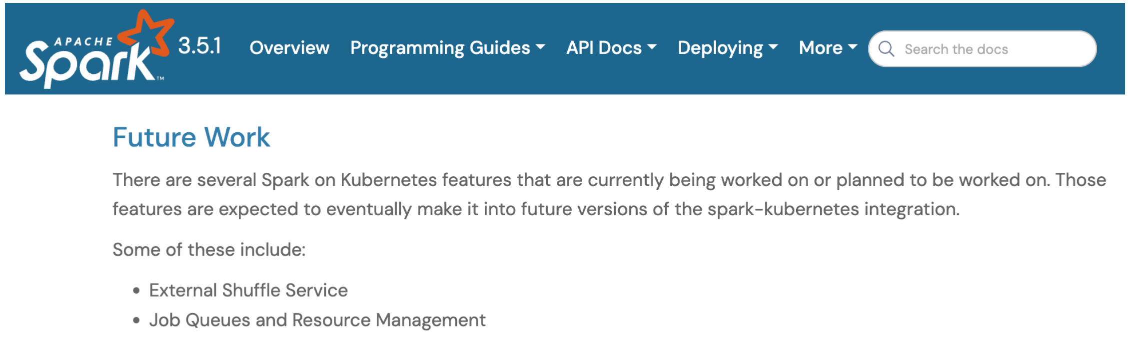
2) We should make a structured streaming algorithm that's separate from the batch algorithm. Eventually, continuous processing might need its own algorithm.

3) Spark should print a warning if you run a structured streaming job when Core's dynamic allocation is enabled

# Scalability: Shuffle data

External shuffle service for Spark on kubernetes is not supported yet. There are 4 options (+1):

- Cloud Shuffle Storage Plugin for Apache Spark - AWS Glue
- IBM/spark-s3-shuffle: Shuffle plugin for Apache Spark and S3 compatible service
- GitHub - oap-project/remote-shuffle: Spark shuffle plugin for support shuffling data through a remote Hadoop-compatible file system (Intel)
- Apache Spark on Kubernetes - Local Storage (main project)
- AWS S3 CSI driver and High-Performance Storage – S3 Express One Zone, AWS




Future Work

There are several Spark on Kubernetes features that are currently being worked on or planned to be worked on. Those features are expected to eventually make it into future versions of the spark-kubernetes integration.

Some of these include:

- External Shuffle Service
- Job Queues and Resource Management

IBM Spark S3 Shuffle plugin is our choice:

- Supports Spark versions 3.2 to 3.4. Successfully tested with Spark 3.5.
- To support different cloud vendors, the corresponding Hadoop connector needs to be added to the classpath.

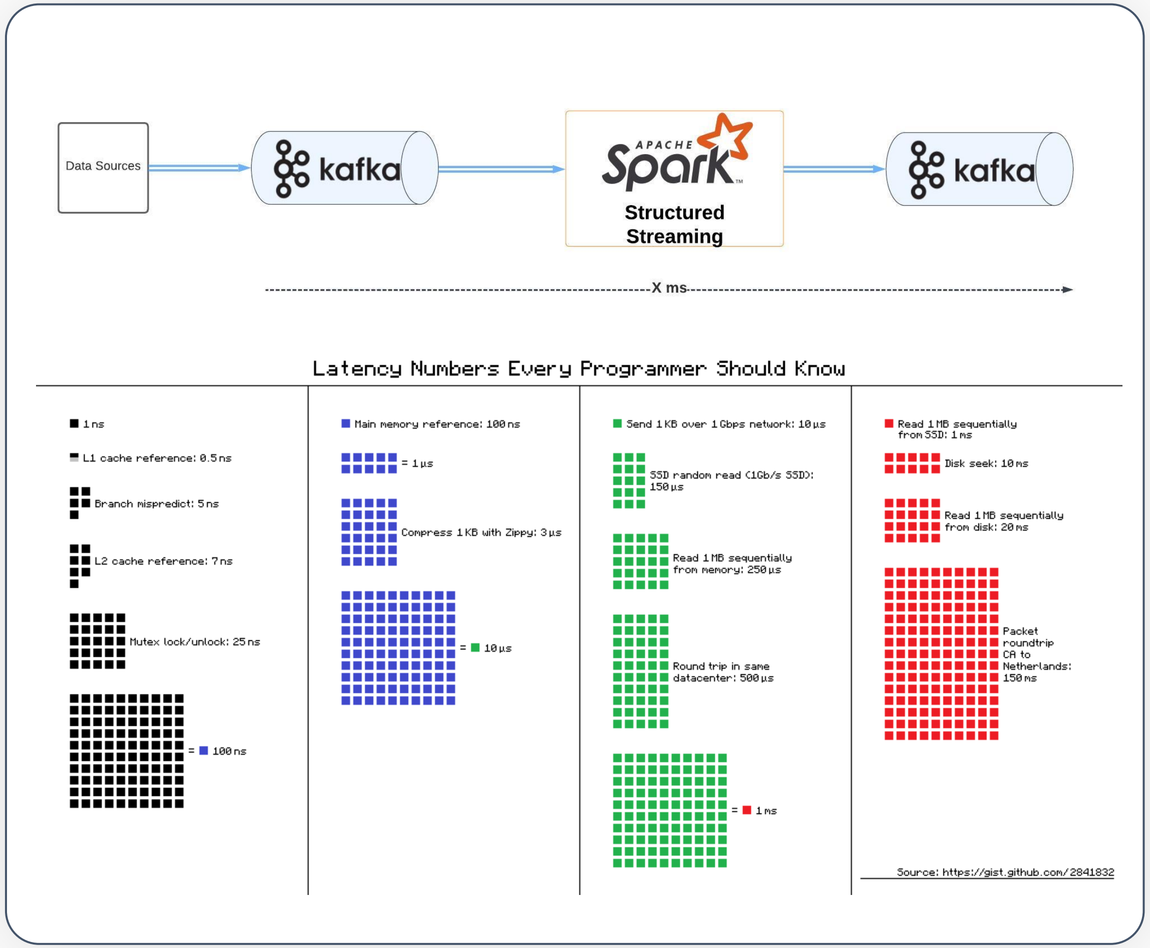
# Latency

## Key factors to consider

- Spark configurations.
- Sub-second latency expectations are a challenge
- Stateful vs Stateless Pipelines

## Consider the following:

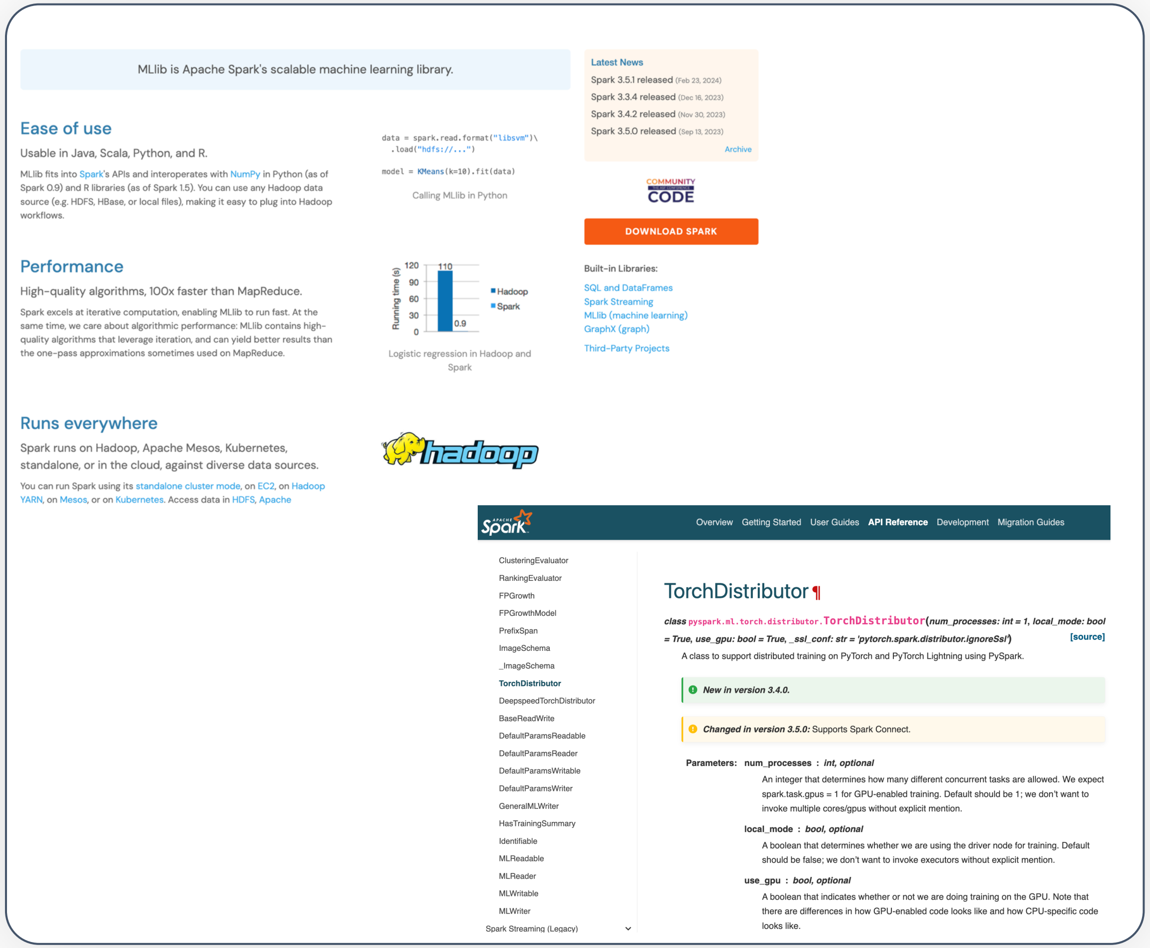
- Use only simple computations involving data transformation or enrichment !
- Always use a message bus (e.g., Apache Kafka or Apache Pulsar) and fast key-value stores (e.g., Apache Cassandra or Redis)
- RocksDB state store provider



# M-L Training

## Key factors to consider

- Most ML frameworks were designed for single-node environments
- Spark MLlib is lagging behind !



# M-L Training

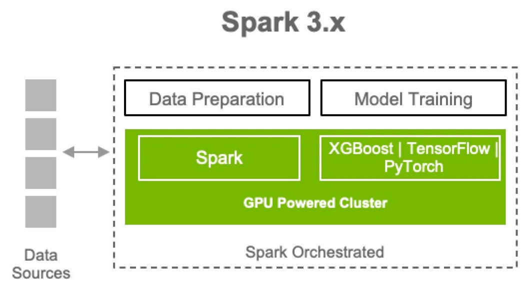
Consider the following:

- Use TensorFlow, Keras, and PyTorch
- Accelerator, Distributed ML & GPU :
  - Horovod
  - NIVIDIA RAPIDS Accelerator for Spark

Distributed training framework for TensorFlow, Keras, PyTorch, and Apache MXNet.

A Unified AI framework for ETL + ML/DL

A single pipeline, from ingest to data preparation to model training spark3cluster




**Source:** [RAPIDS Accelerator for Apache Spark](RAPIDS Accelerator for Apache Spark) (NVIDIA)

# M-L Training (Batch)

With the default scheduler: workloads experience higher rates of resource starvation, leading to performance degradation or failure !

Solutions for the default scheduler:

- Use custom k8s Scheduler support. Enabling YARN-like capabilities such as queue, gang scheduling, etc

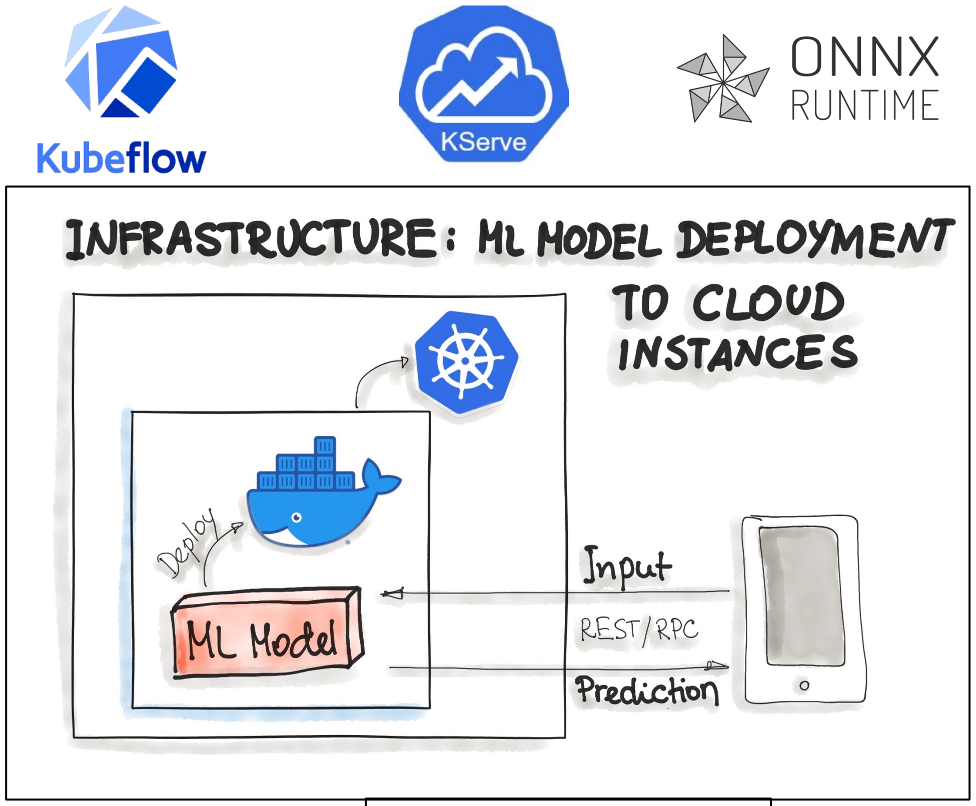
VOLCANO
Kubernetes Native Batch System

Apache YuniKorn

# M-L Deployment & Serving (WIP)

## Key factors to consider

- Package the whole ML tech stack (dependencies) and the code for ML model prediction into a Docker container.
- Model optimization and Model compression




Source: https://ml-ops.org/

# Conclusion and takeaways

# Why Spark on k8s integration is Important for R-L M-L?

- Native Integration
- k8s best practices apply to Spark on k8s for free!
- Scalability, Latency, Fault-tolerance
- Models Training and Serving
- Integration with a rich ecosystems

# Key takeaways

- Use the k8s Spark operator
- Design and build your logging and monitoring stack
- Keep adhering to Spark best practices compatible with k8s
- Use the rich k8s (Monitoring, Mlops, etc) ecosystem
- Contribute to the OSS (share your experiences, code, ideas, challenges )
- Keep Cycling …

# Thank you

Blogs

Ocean for Apache Spark

instaclustr
Now part of **Spot by NetApp**