COWBOY DATING WITH BIG DATA

DATA PLATFORM EVOLUTION IN ACTION

BORIS TROFIMOV

ABOUT ME

Big Data competence lead @ Sigma Software

Worked with Verizon/Yahoo/AOL, Collective

Cofounder of Odessa JUG

Passionate follower of Scala

Associate professor at ONPU

SIGMA
Expect a better tomorrow

INTRO – PARTNER 1

INTRO – PARTNER 2

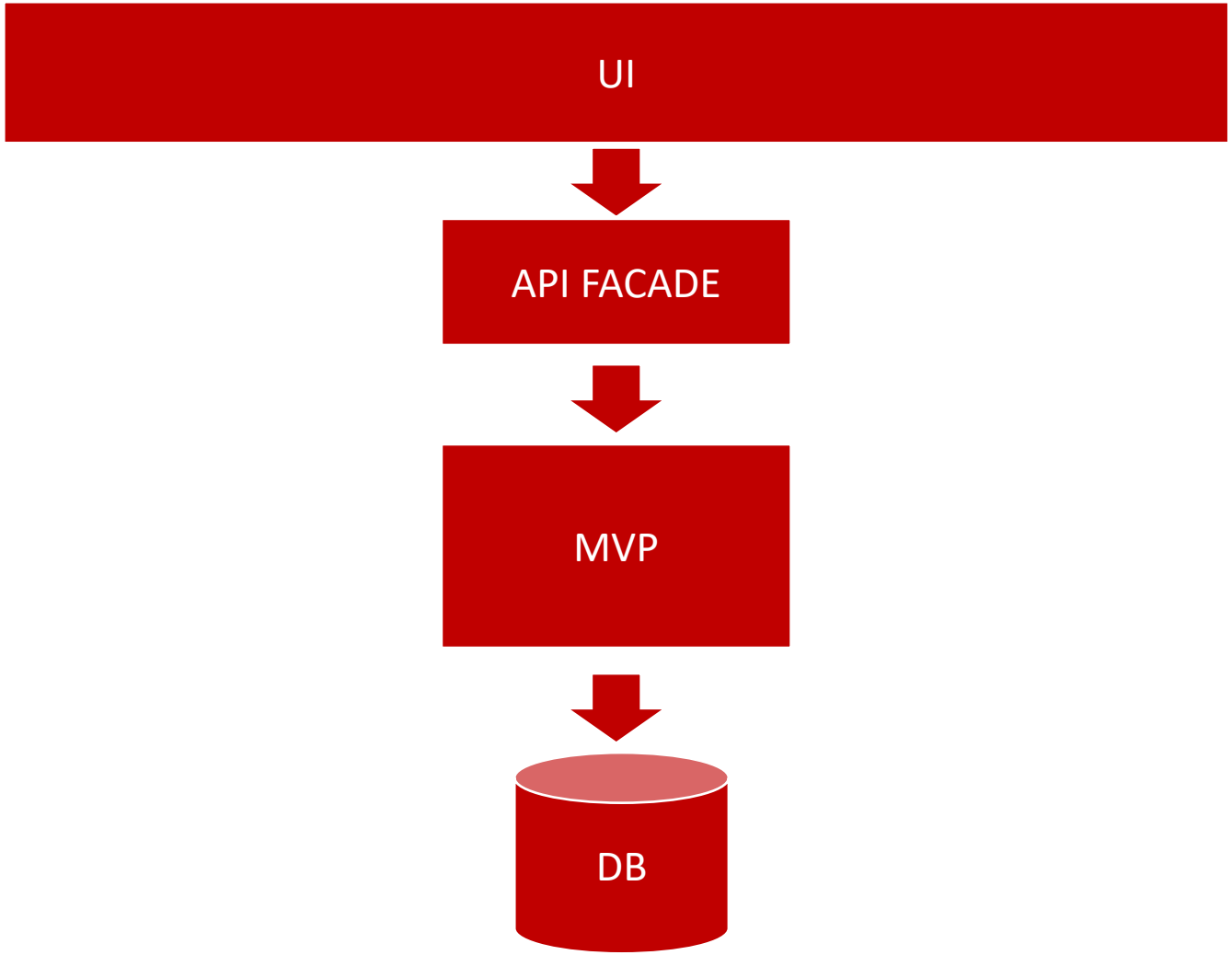
НАГРАДЫ PARTNER 2

- Медаль за Kotlin
- Полный кавалер Spring & Spring Boot
- Орден за взятие Kubernetes

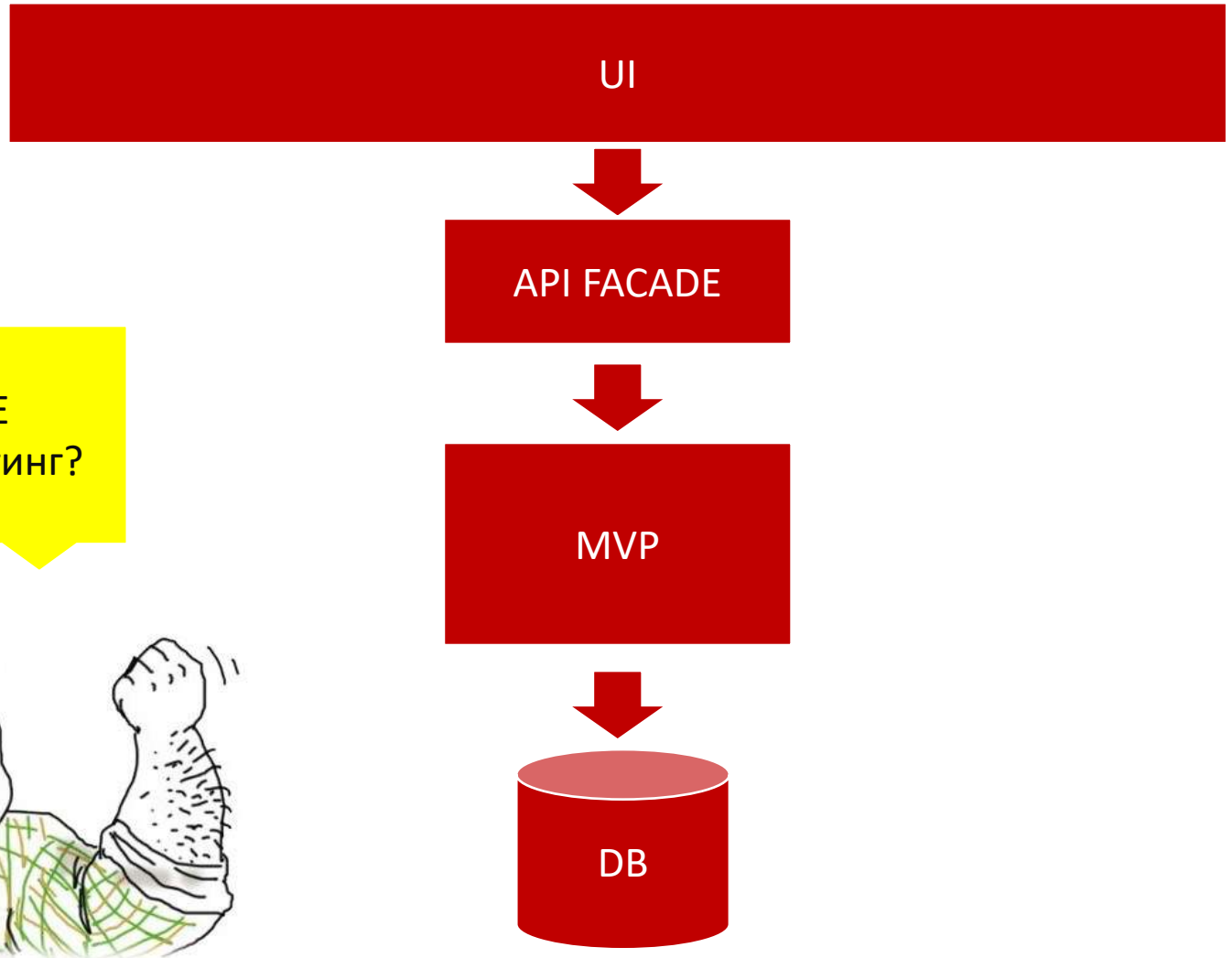
LESSON 1 – SHARED STORAGE

EXPECTATIONS

SHARED STORAGE

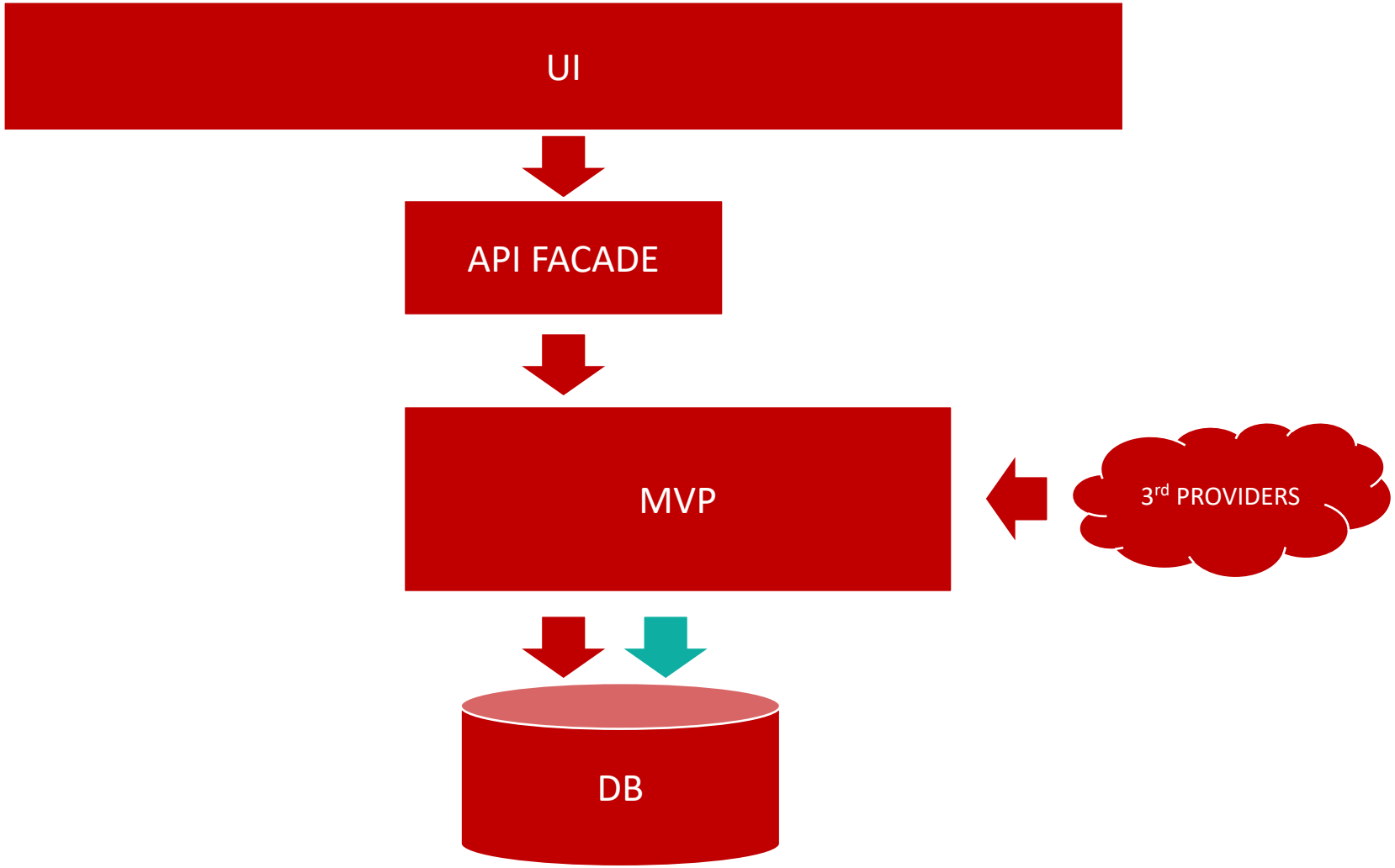


SHARED STORAGE



ГДЕ
репортинг?

SHARED STORAGE



SHARED STORAGE

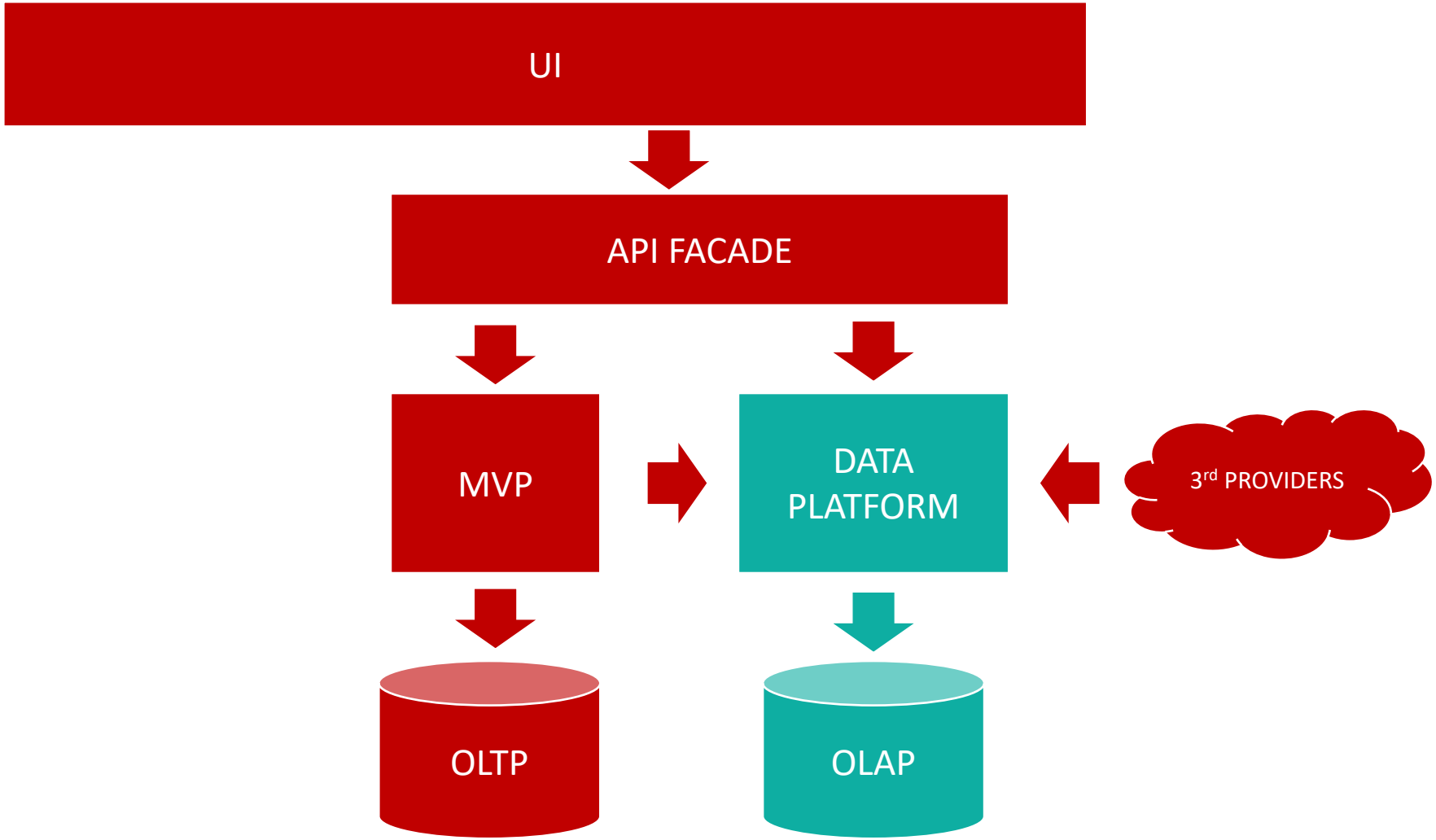
PROS

- Fast TTM
- Relatively cheap from infra and cost perspective

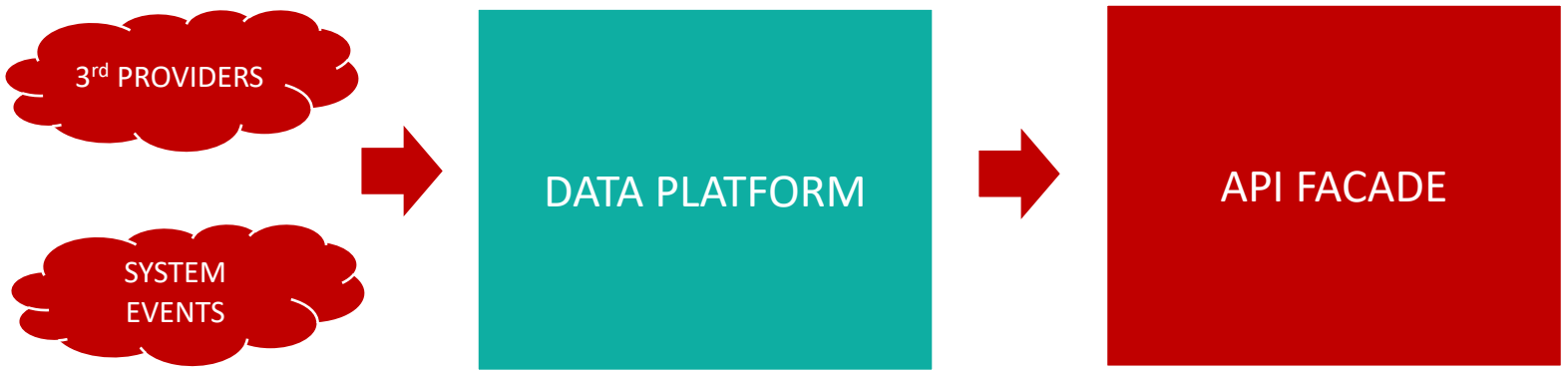
CONS

- Tight data and code cohesion
- Different Scaling scenarios
- Performance and Availability issues

SHARED STORAGE

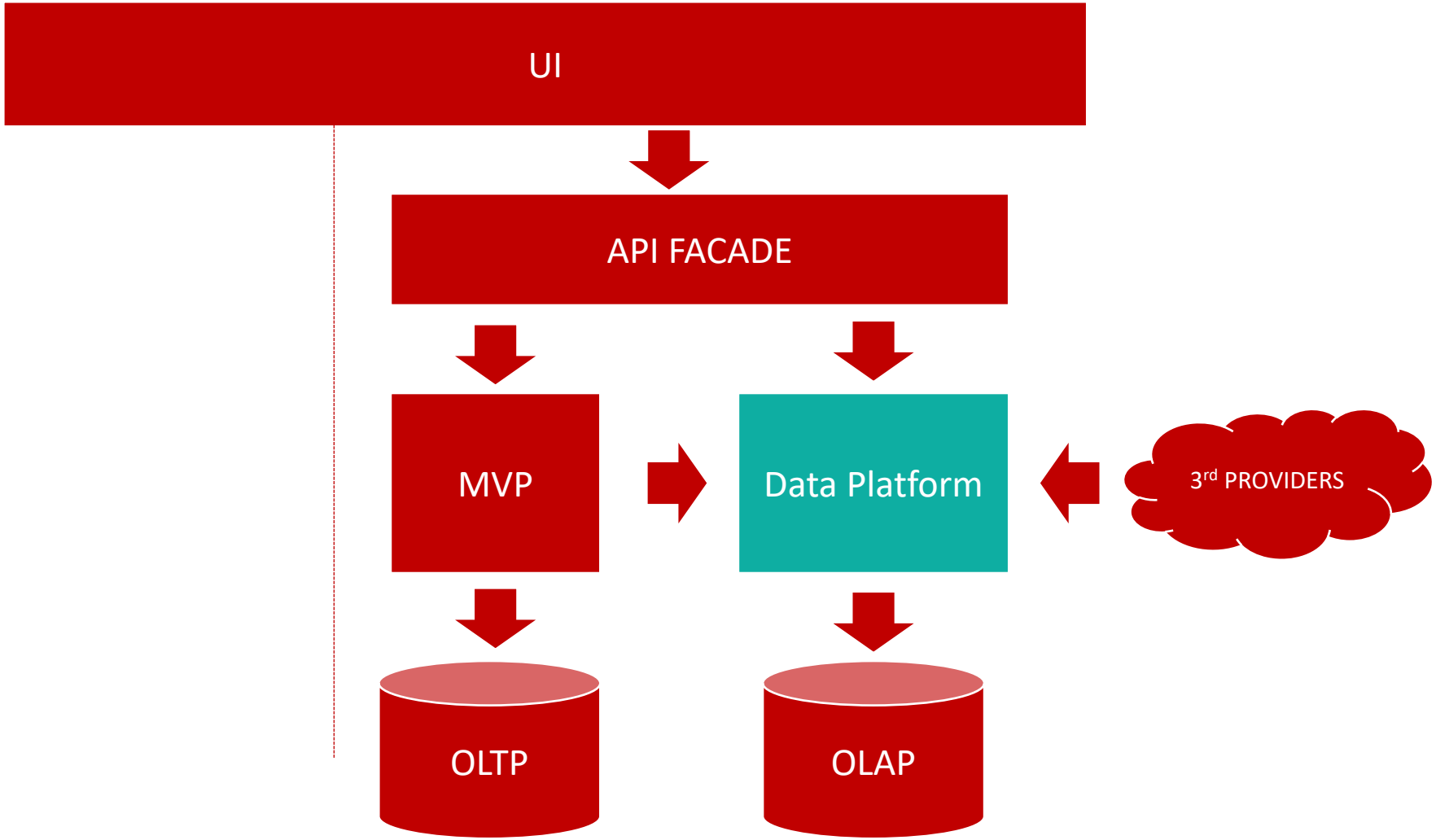


DATA PLATFORM

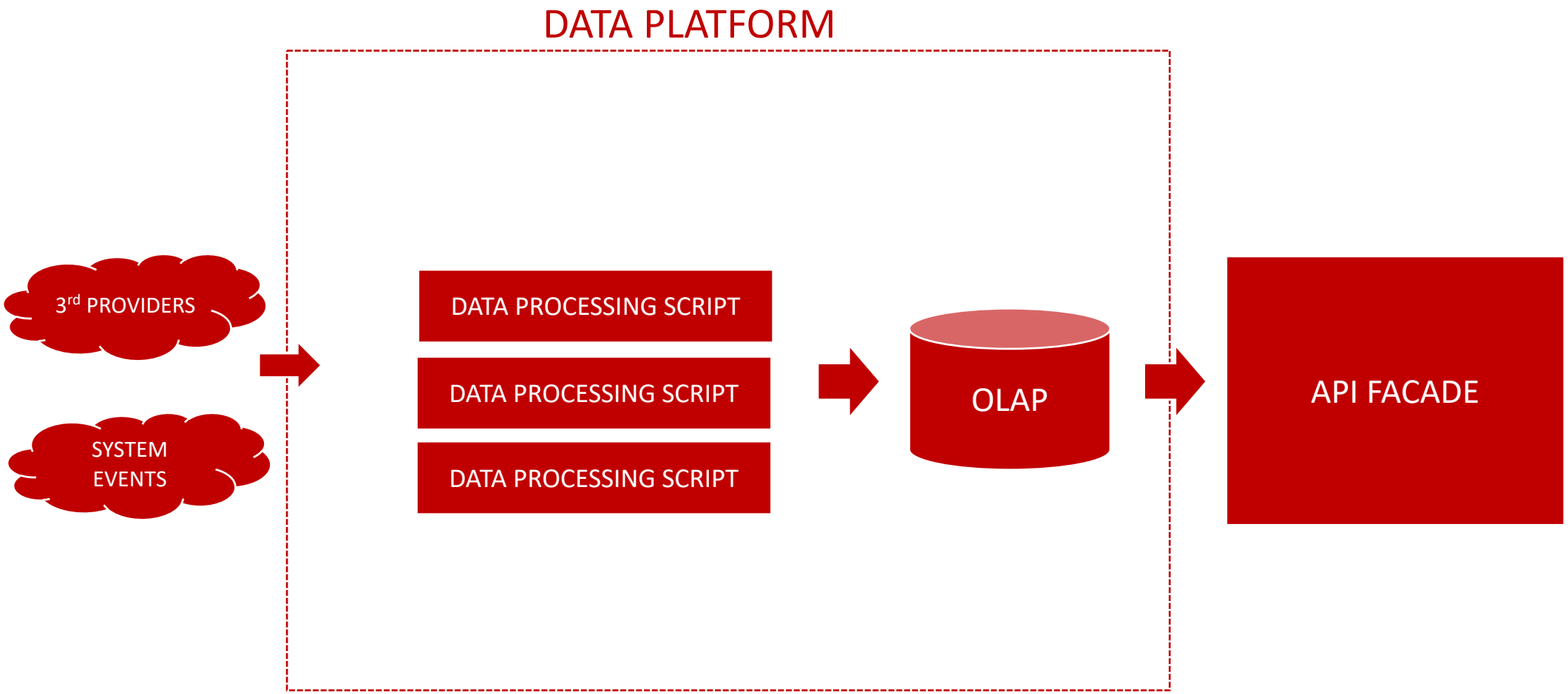


LESSON 2 – WEAK SCHEDULING

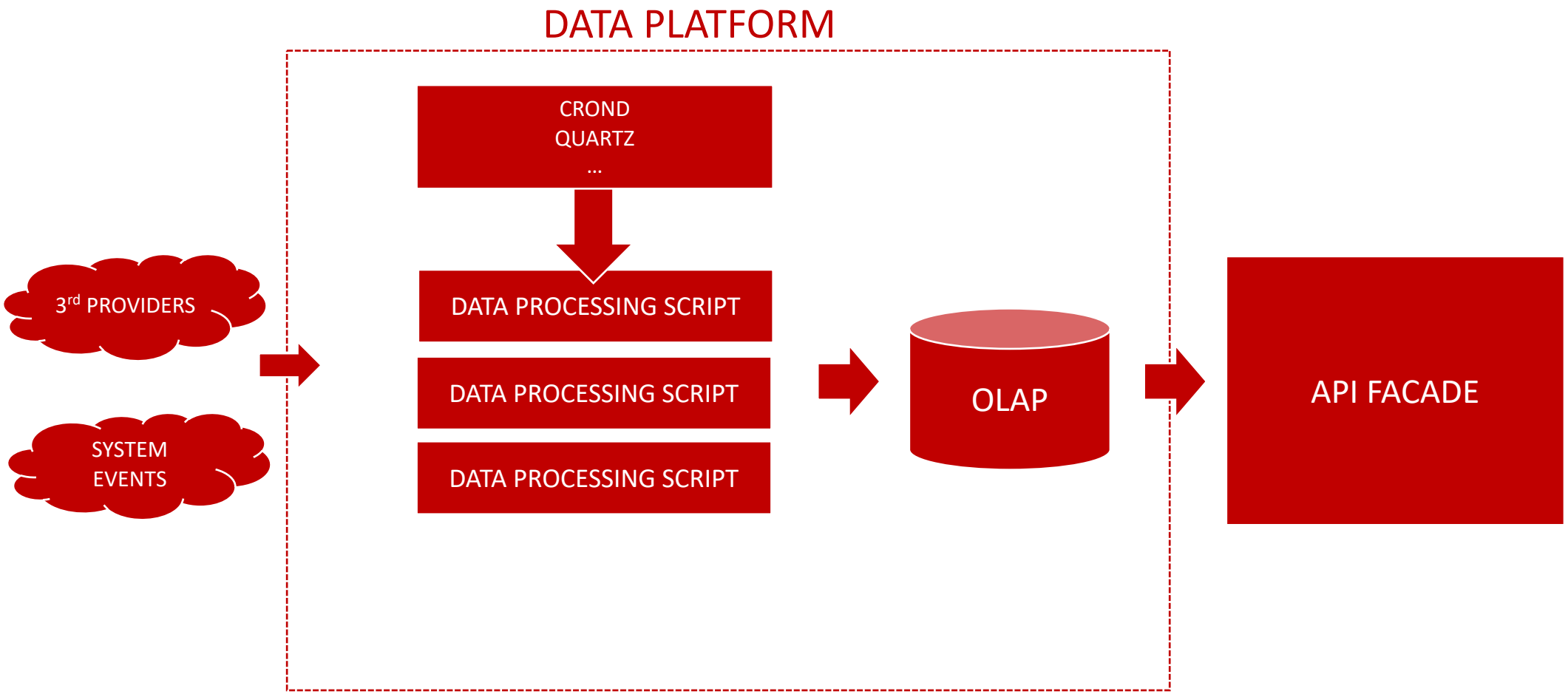
SCHEDULING



SCHEDULING

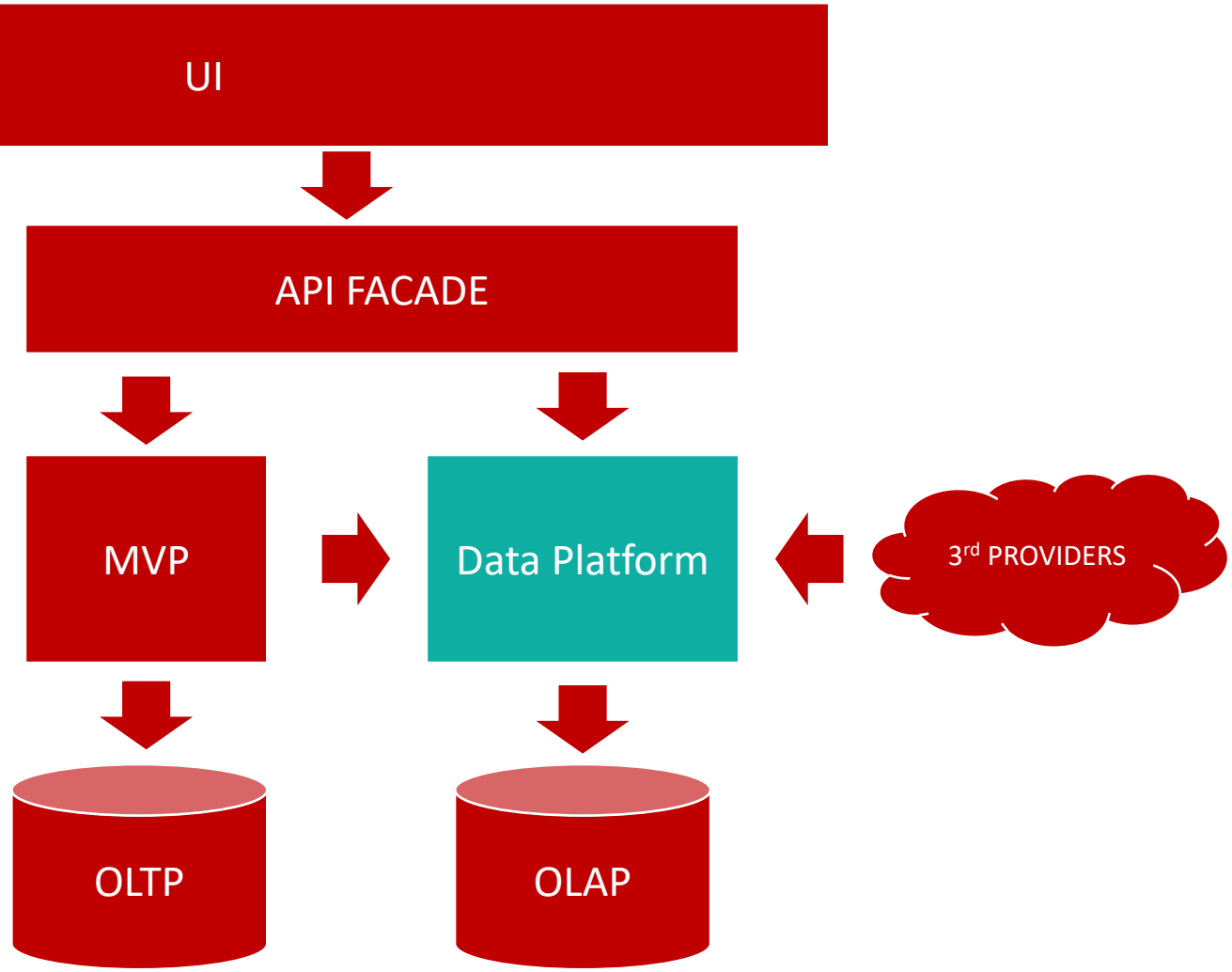

DATA PLATFORM
3rd PROVIDERS
SYSTEM EVENTS
DATA PROCESSING SCRIPT
DATA PROCESSING SCRIPT
DATA PROCESSING SCRIPT
OLAP
API FACADE

SCHEDULING


DATA PLATFORM
CROND
QUARTZ
...
DATA PROCESSING SCRIPT
DATA PROCESSING SCRIPT
DATA PROCESSING SCRIPT
3rd PROVIDERS
SYSTEM
EVENTS
OLAP
API FACADE

SCHEDULING


КАКОЙ
КРОНТАБ?


UI
API FACADE
MVP
Data Platform
3rd PROVIDERS
OLTP
OLAP

SCHEDULING

Use prod-ready schedulers

- Airflow
- Azkaban
- Oozie

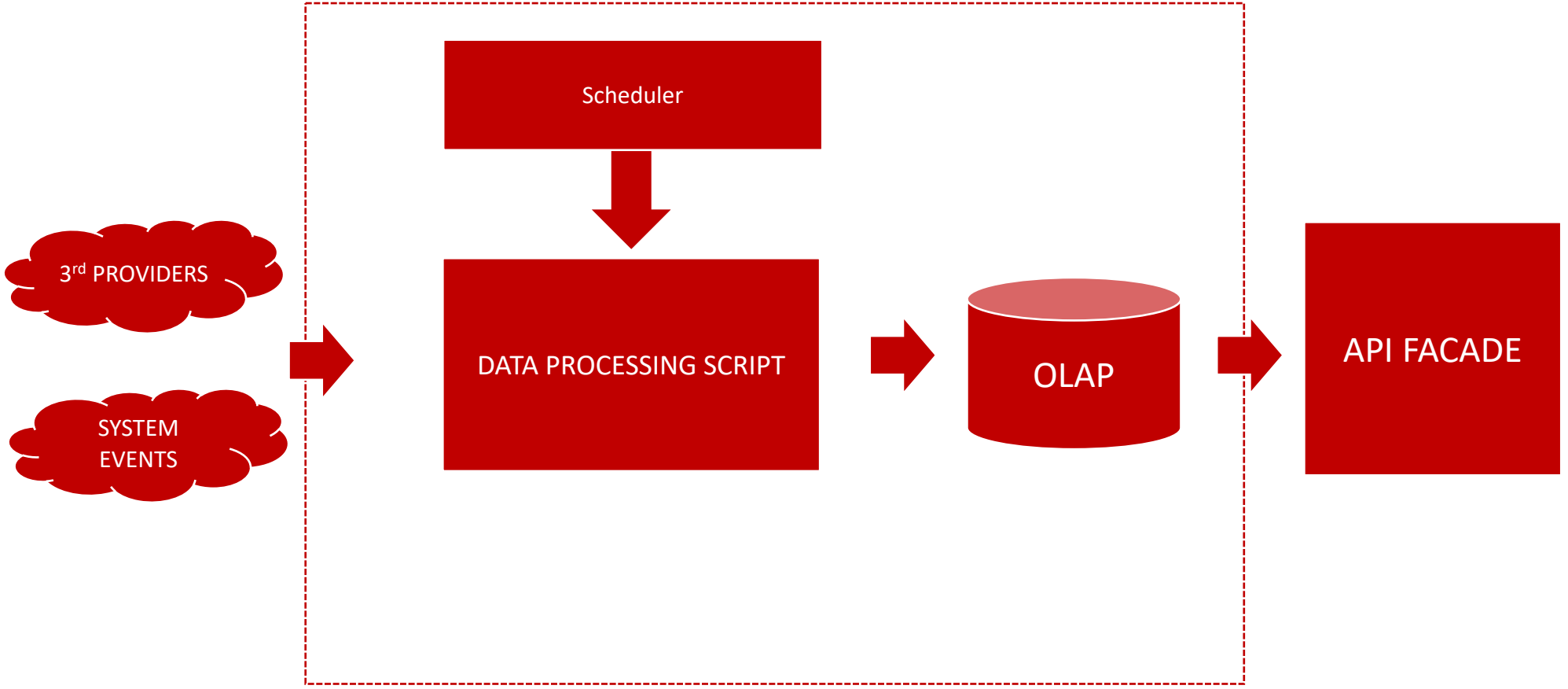
- Jenkins?

What we gain

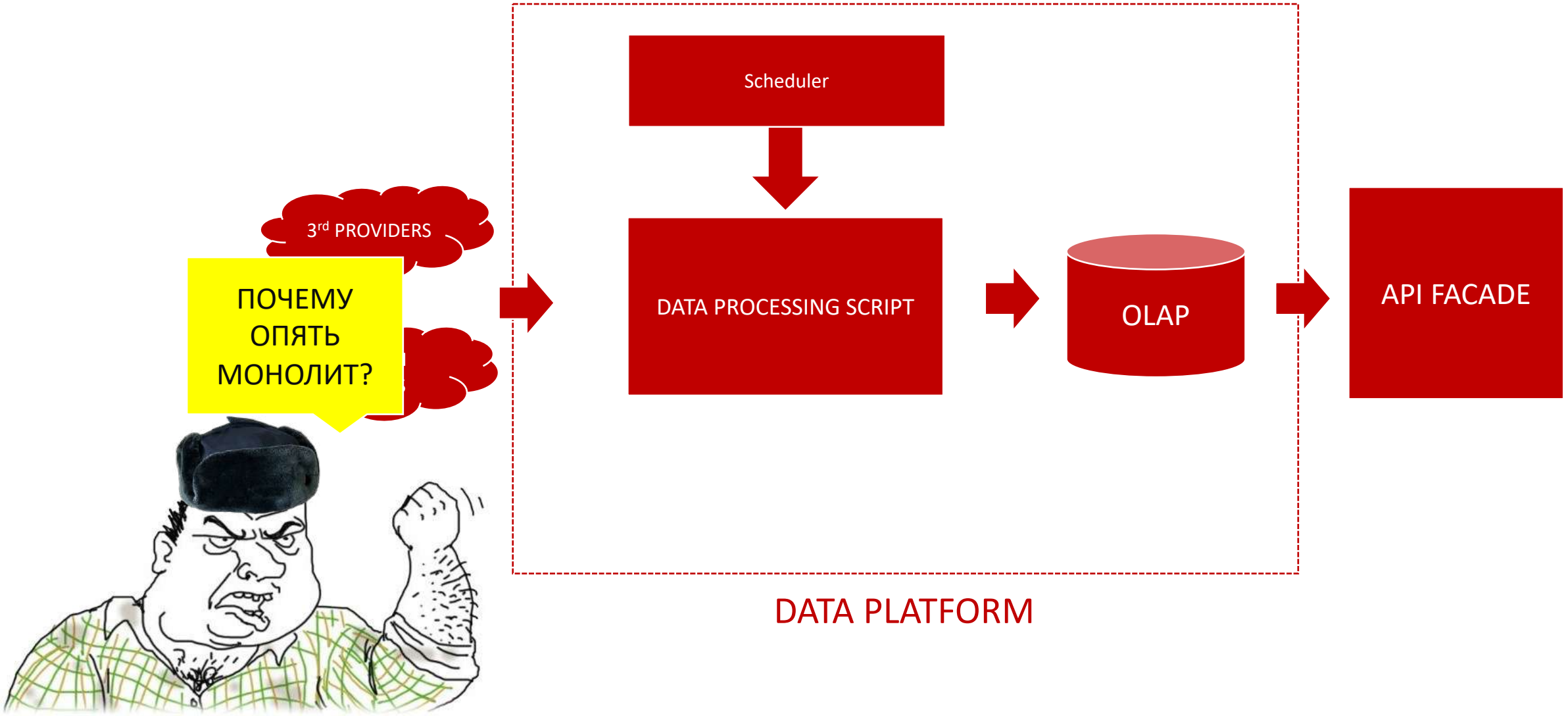
- Identity control and Audit
- Job Lineage, Logging and Troubleshooting
- Tools to design Workflows/DAGs
- Fault Tolerance features (rerun etc.)

CHAPTER 3 –MONOLITHIC DATA PLATFORM

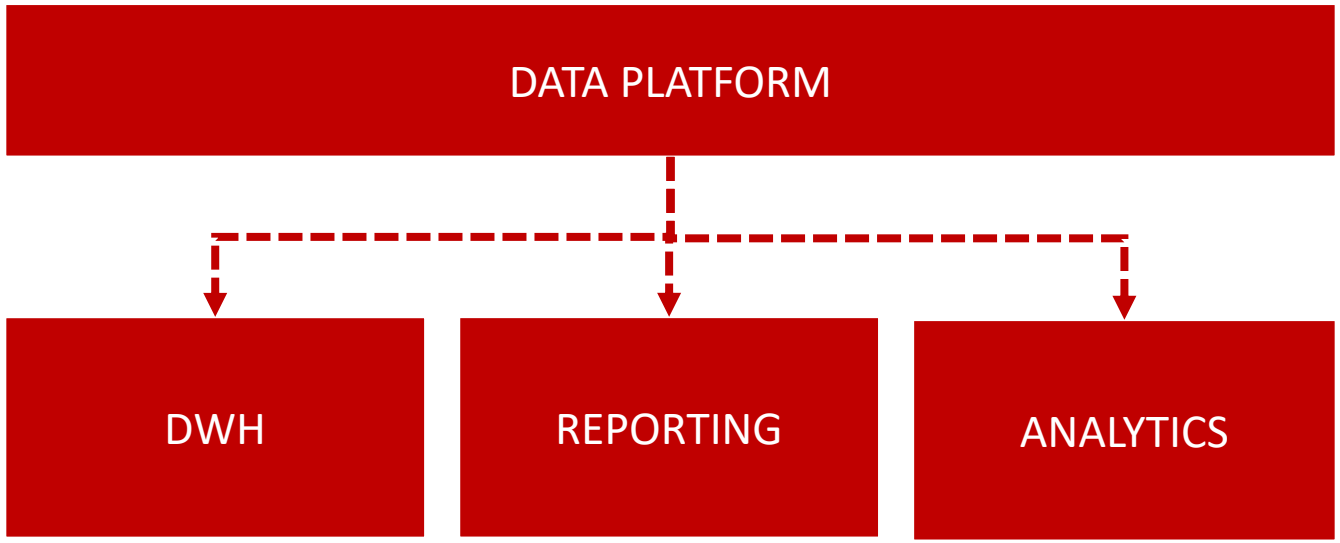
DATA PLATFORM



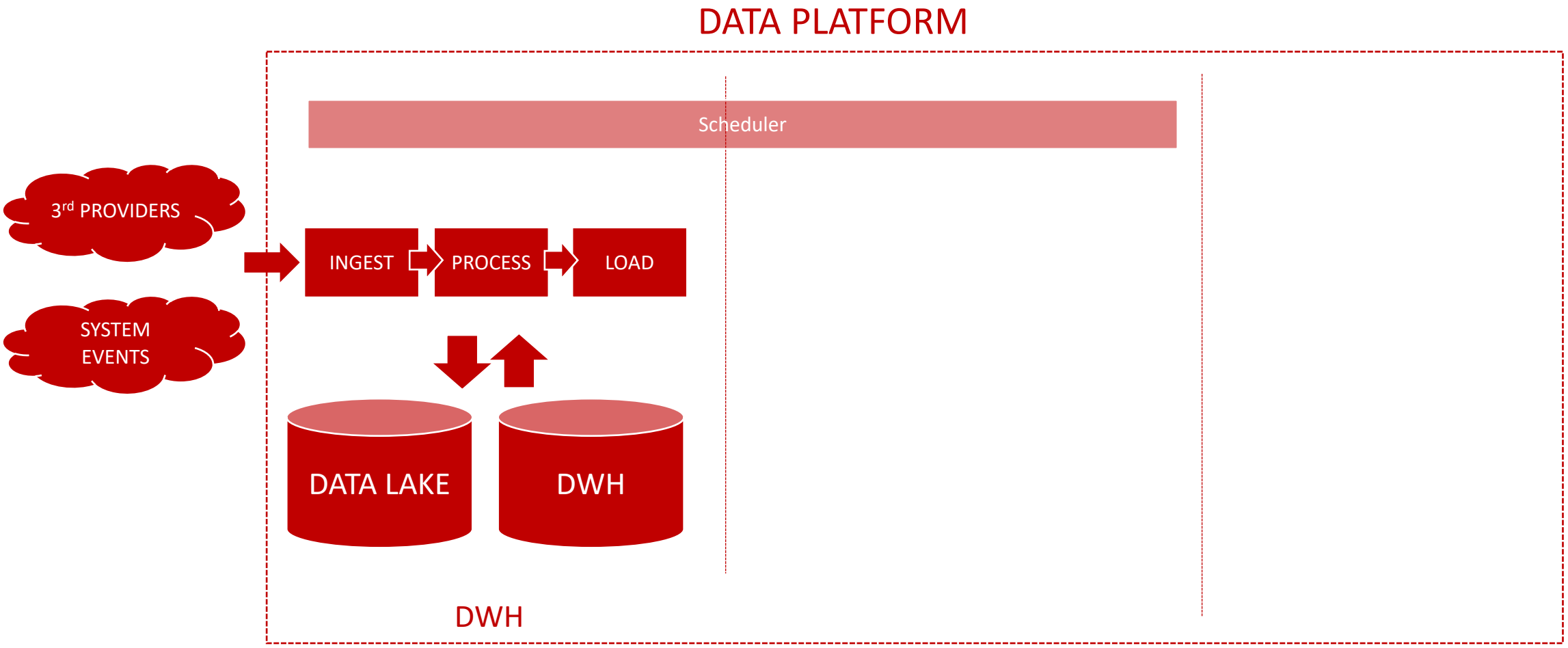
DATA PLATFORM



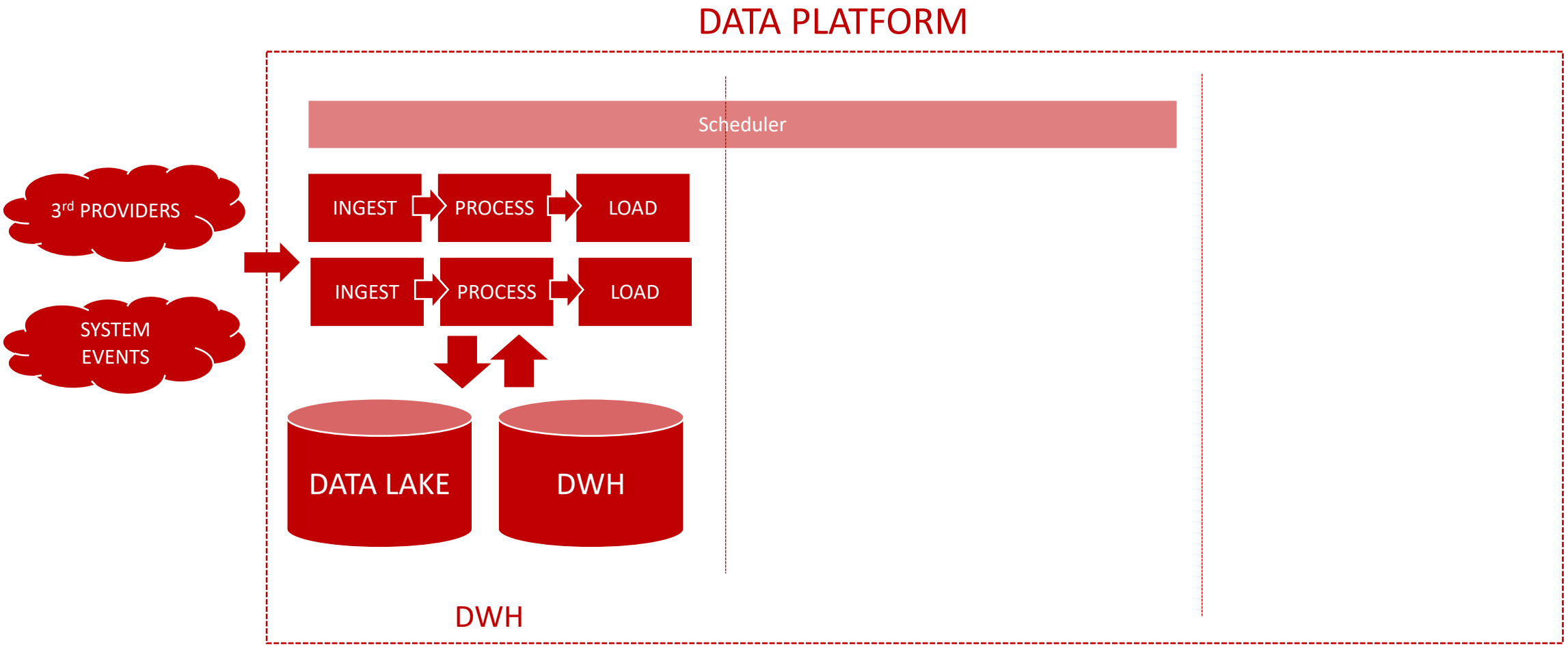
DECOUPLING DATA PLATFORM



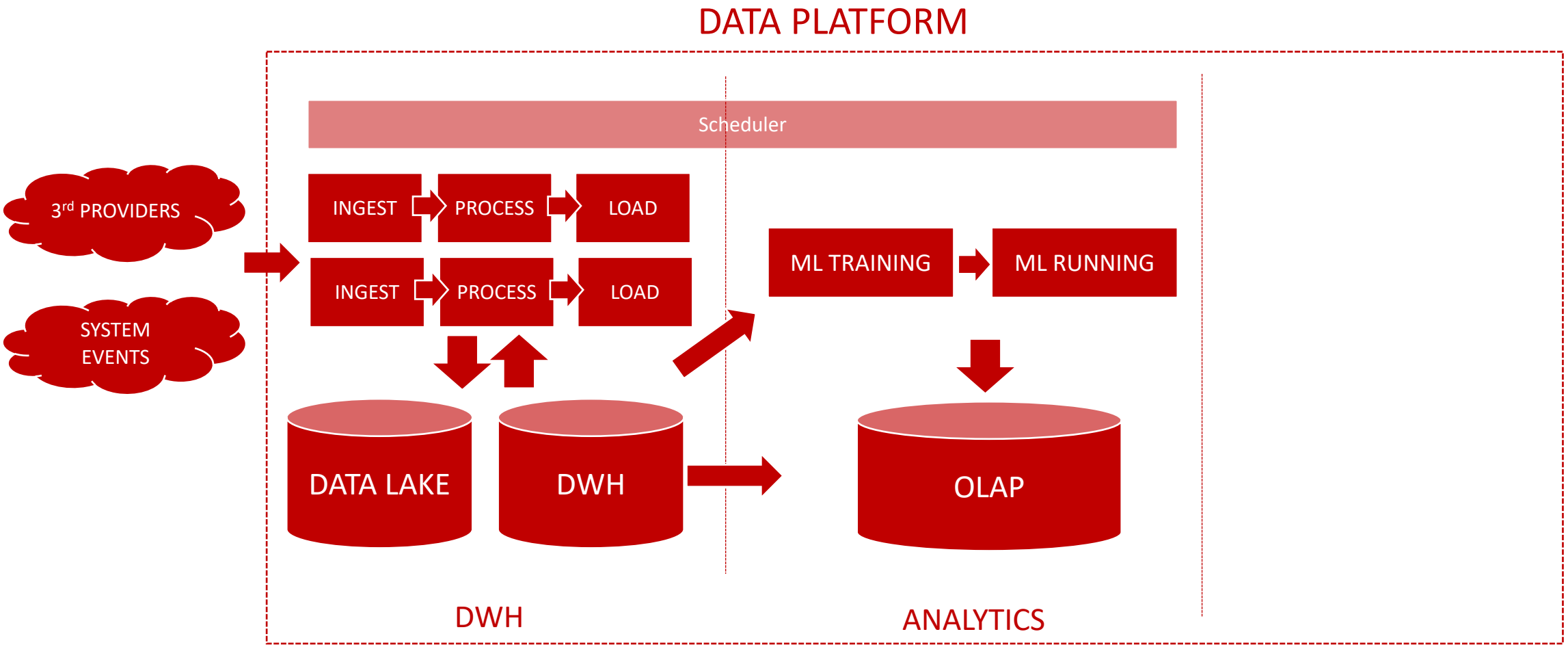
DECOUPLING DATA PLATFORM



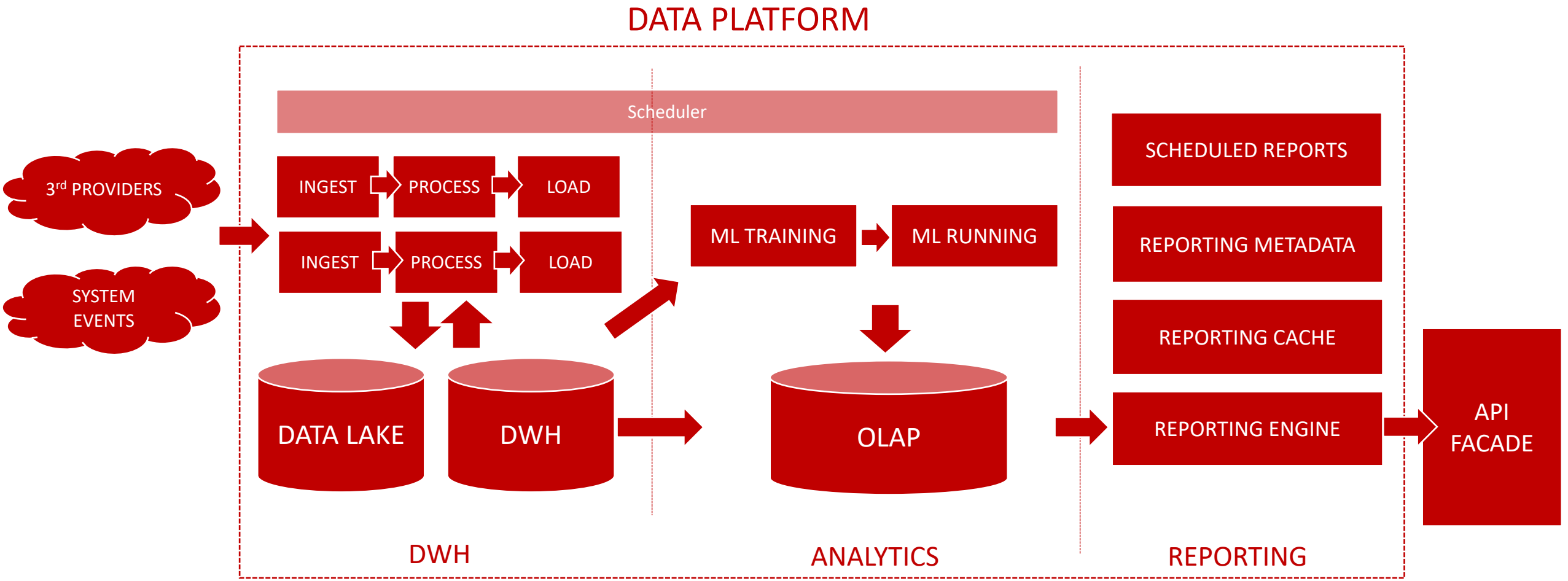
DECOUPLING DATA PLATFORM



DECOUPLING DATA PLATFORM



DECOUPLING DATA PLATFORM



DECOUPLING DATA PLATFORM

ARCHITECTURE DECISIONS

- Raw data should be stored inside Data Lake
- Introduce granular reusable and testable steps inside pipelines
  [ingest, validate, enrich, aggregate etc.]
- Separate pipeline per vendor/feed
- Introduce Data Linage, easy troubleshooting
- Separate concerns (Scalability, Fault Tolerance)

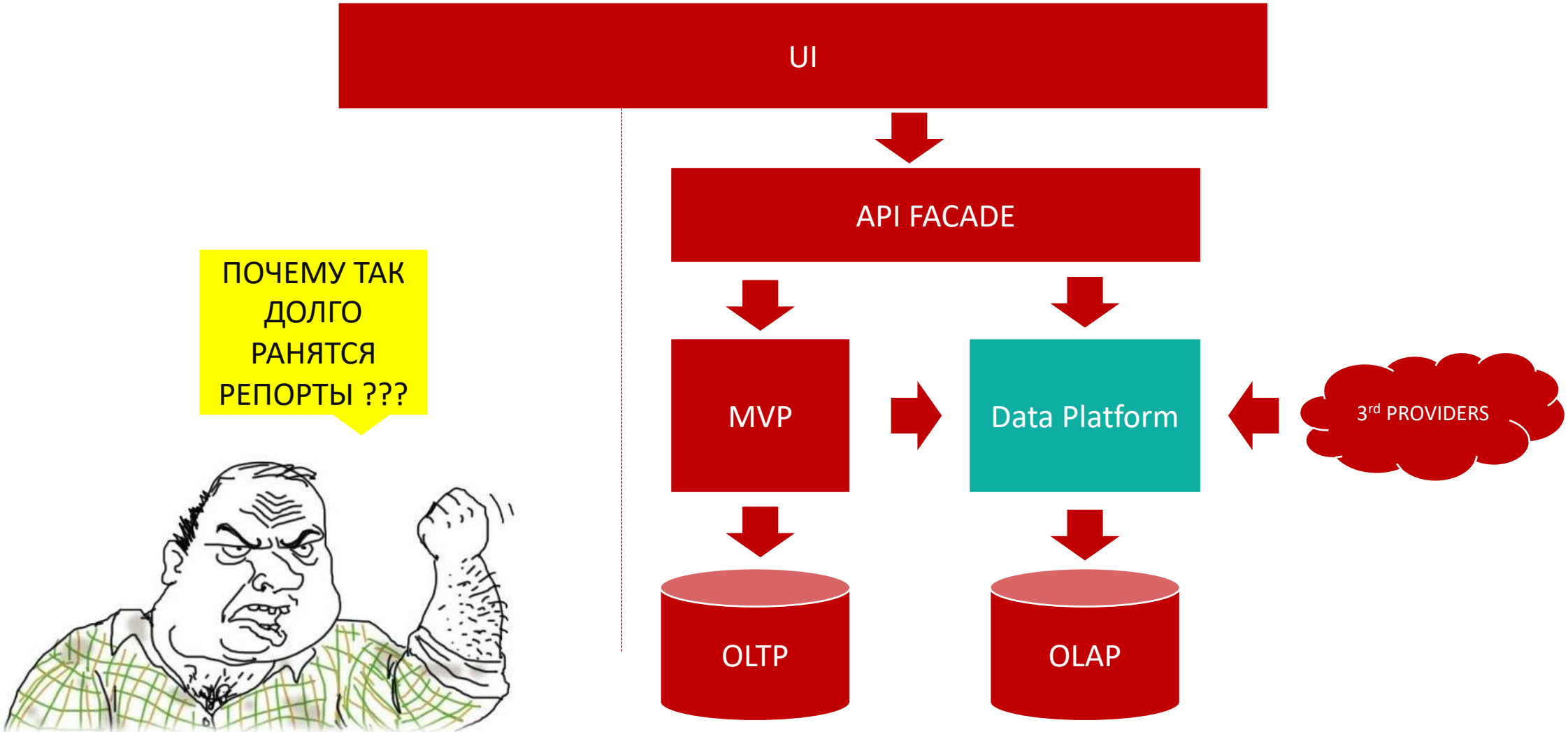
DECOUPLING DATA PLATFORM

VENDOR-AGNOSTIC TECHNOLOGY STACK

- Apache NiFi for data routing and ingestion
- Apache Spark/Flink/Presto/Beam for processing
- Kafka/Hive for Data Lake Storage
- Hive/Memsql for DWH
- Vertica/Redshift/Memsql/Clickhouse for OLAP

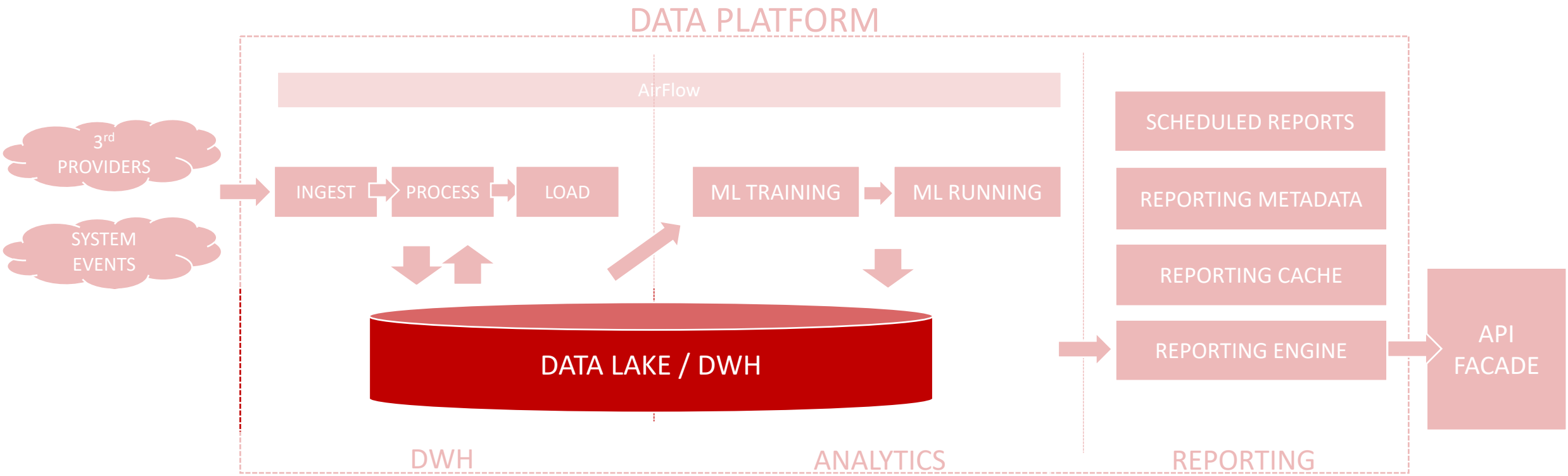
LESSON 4
AGGREGATE IT

AGGREGATE IT



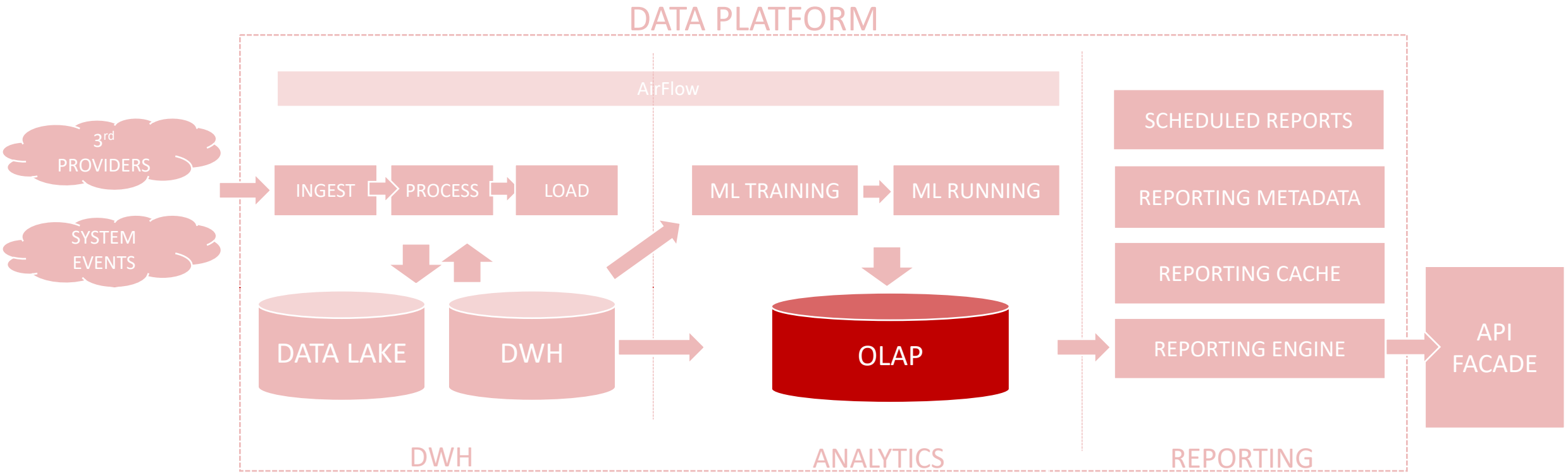
COMMON PITFALLS

- Direct access to Data Lake or cold DWH



COMMON PITFALLS

- Reports query RAW data



INTRODUCE AGGREGATIONS

impressions

date	hour	user_cookie	creative_id
05/21/19	03	4444444	123
05/21/19	03	5555555	321
05/21/19	03	6666666	321
05/21/19	04	7777777	567

campaigns

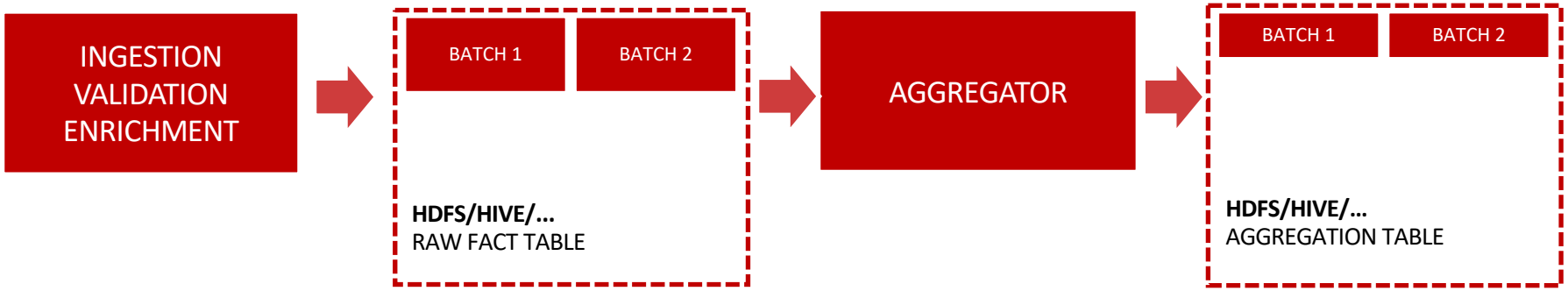
creative_id	campaign_id
123	1
321	1
567	2

JOIN

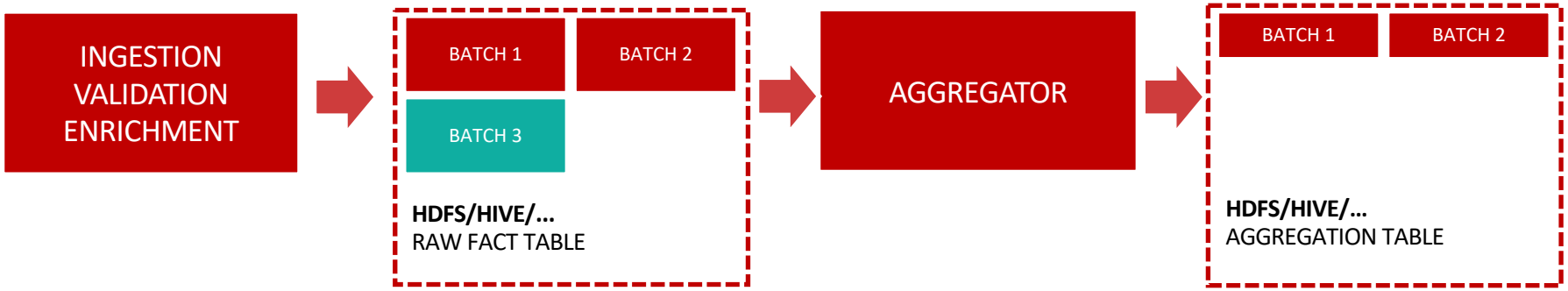
performance_ad

date	hour	campaign_id	creative_id	impressions
05/21/19	03	1	123	1
05/21/19	03	1	321	2
05/21/19	04	2	567	1

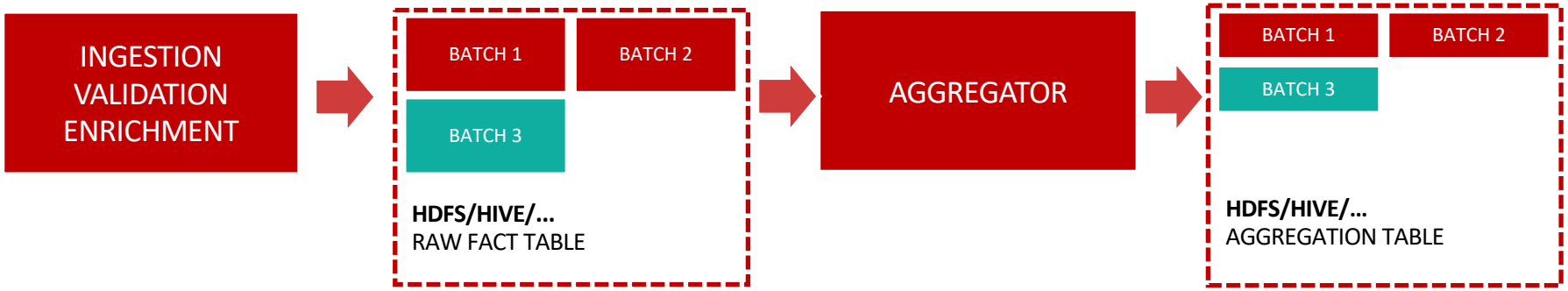
INTRODUCE AGGREGATIONS



INTRODUCE AGGREGATIONS

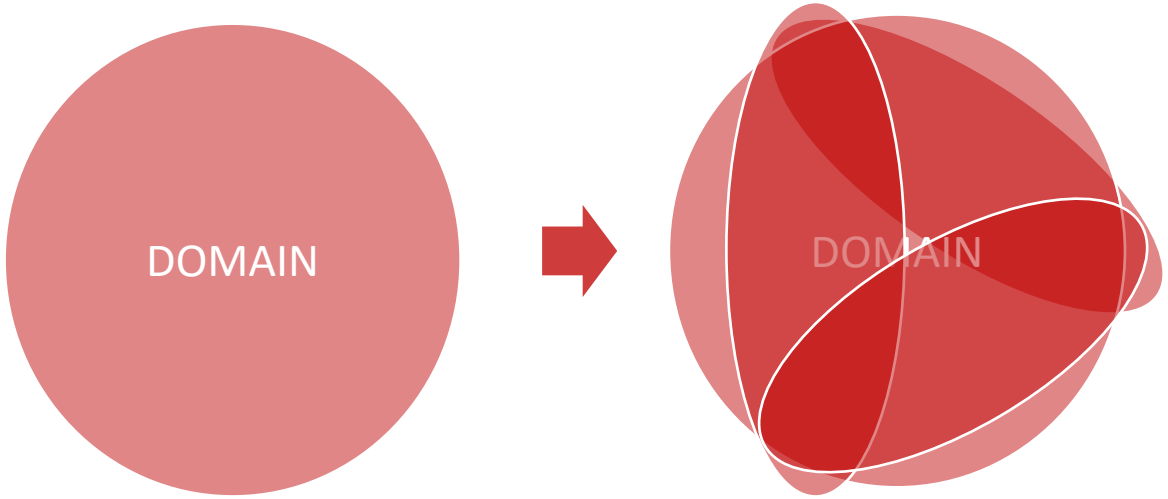


INTRODUCE AGGREGATIONS



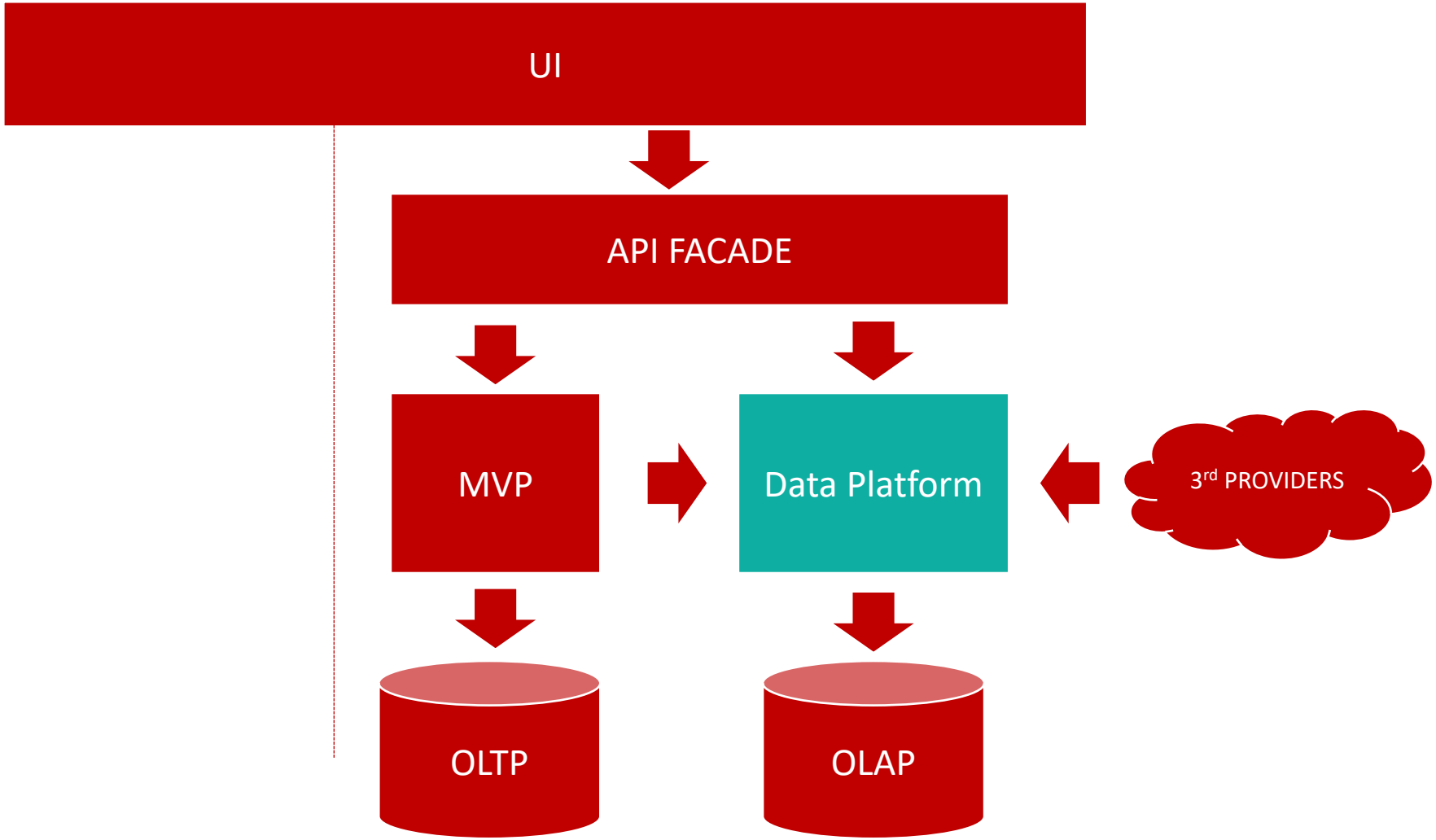
BREAK DOWN DOMAIN

- Break down domain into business-concerned areas
- Cover area with dedicated aggregation
- Example For Video Platform
  - Ad performance
  - Player performance
  - Video performance
  - Revenue performance
- Build once, reuse between multiple reports

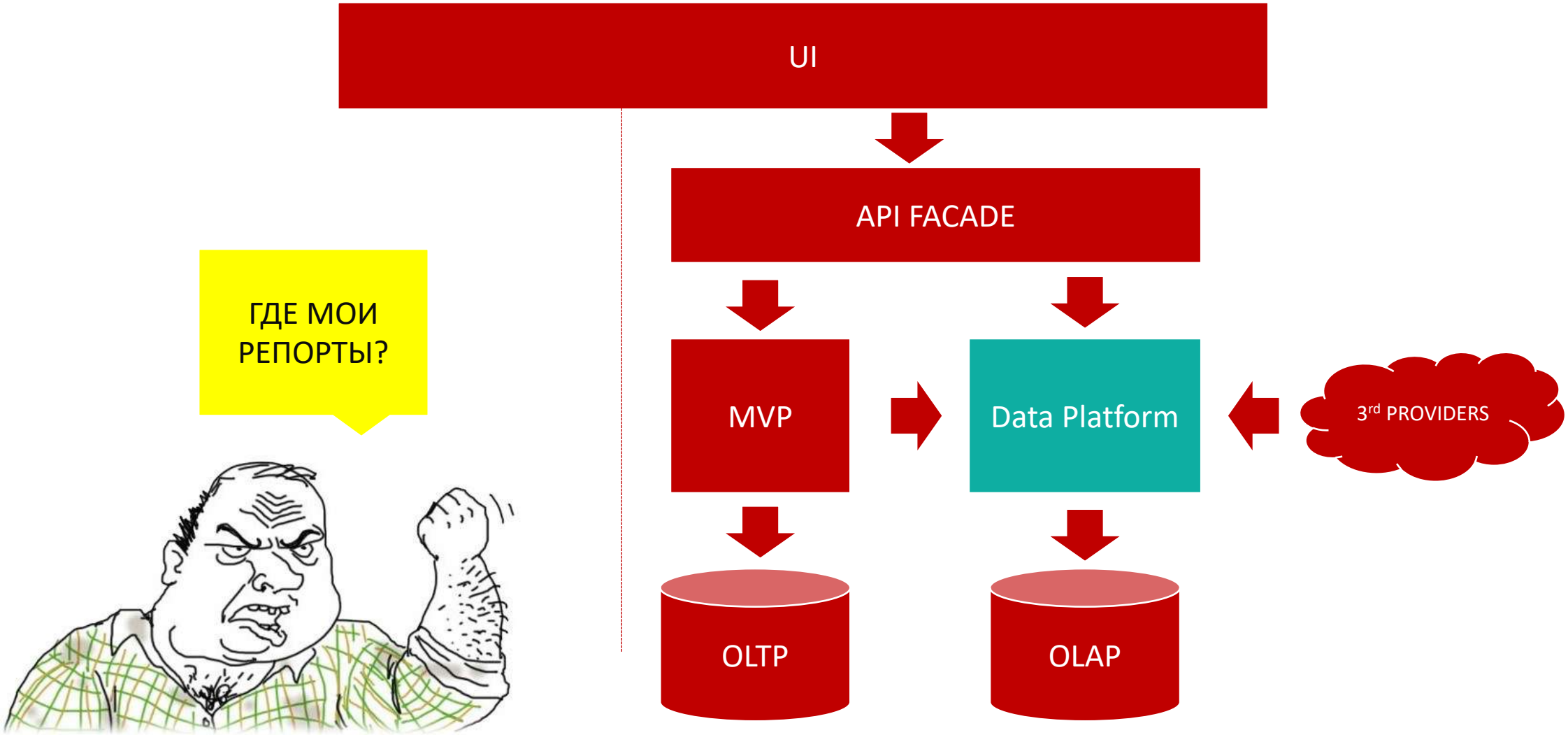


LESSON 5
AVAILABILITY

AVAILIABILITY



AVAILIABILITY



THINGS EASY TO MISS

AVAILABILITY

- If possible do not share infrastructure between DP with Core services
- Chose wise between Kappa and Lambda architectures
- Introduce effective monitoring
- Know your data latency and design solution based on it

THINGS EASY TO MISS

FAULT TOLERANCE

- Every job should be fail-ready and retry-able by design
- Enable multiple attempts on scheduler side
- Use idempotent sinks
- Implement backpressure: Prefer Pull over Push, leverage Blob/S3/HDFS or Kafka

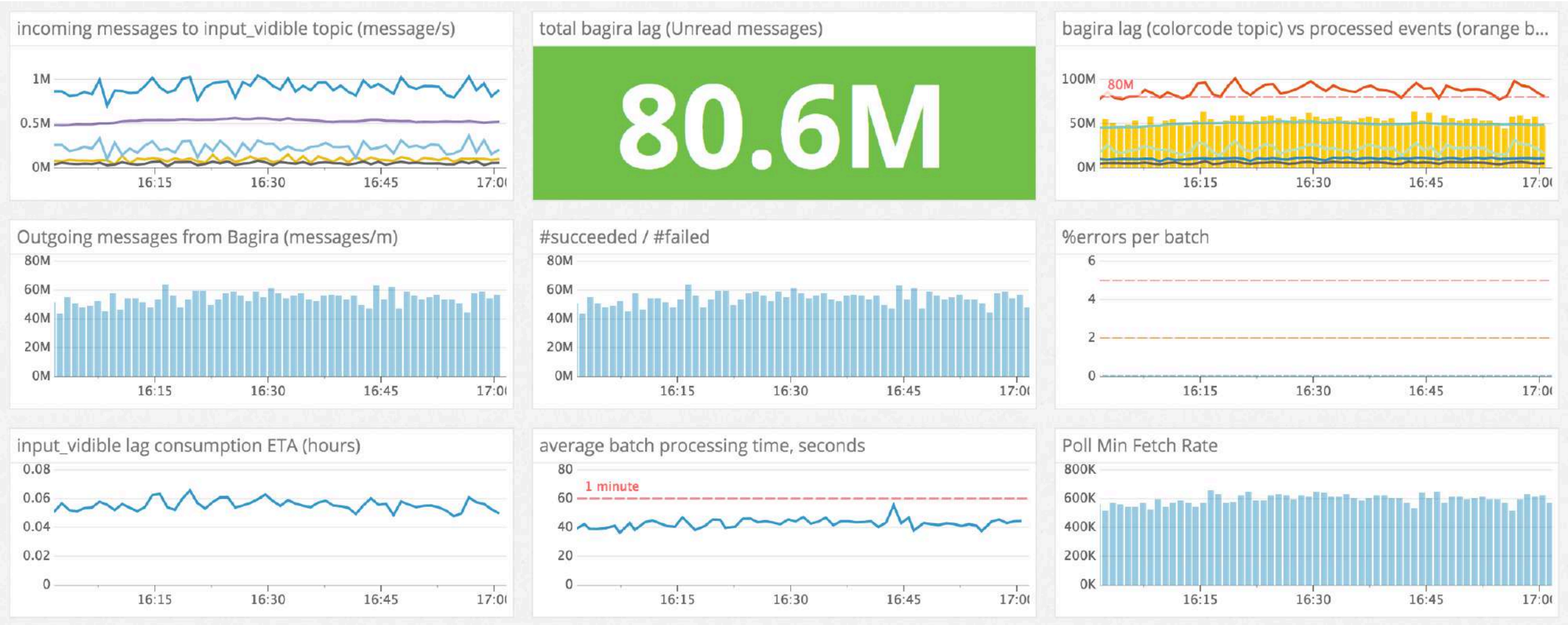
THINGS EASY TO MISS

EFFECTIVE MONITORING

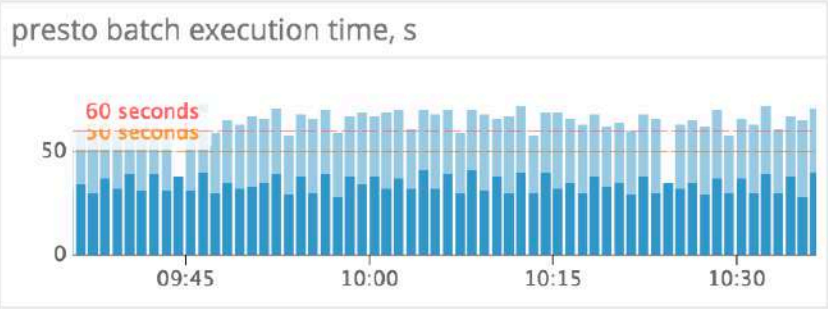
- Collect system and app-specific metrics
- Measure data availability [in-rate, out-rate, lag]

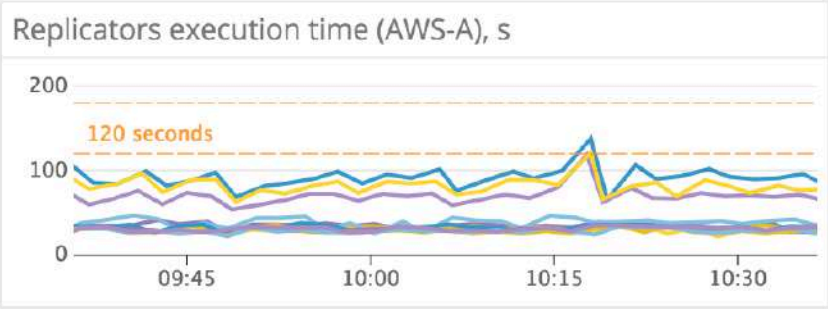
  Bandar-Log https://github.com/VerizonAdPlatforms/bandar-log/
- Think about Datadog [local agents, dashboards, monitors, notes]

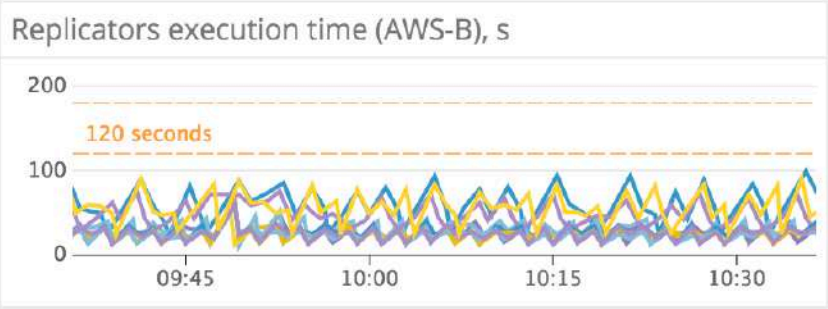
DASHBOARD EXAMPLE [INGESTION]

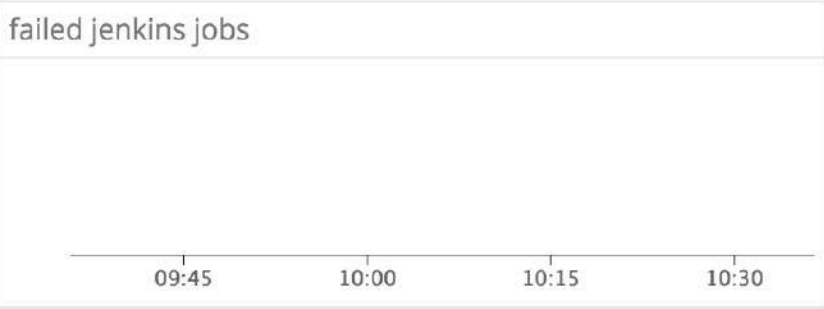


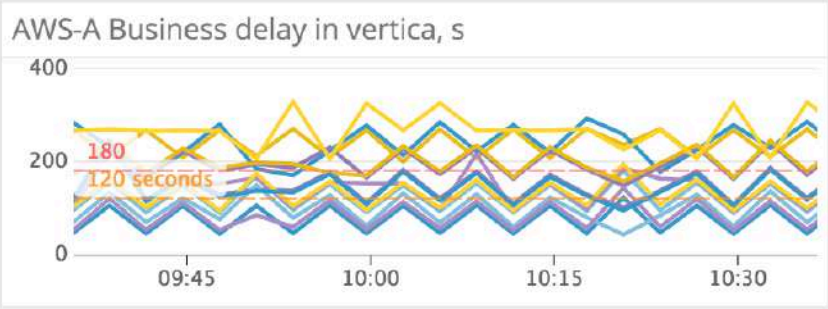
DASHBOARD EXAMPLE [AGGREGATIONS]

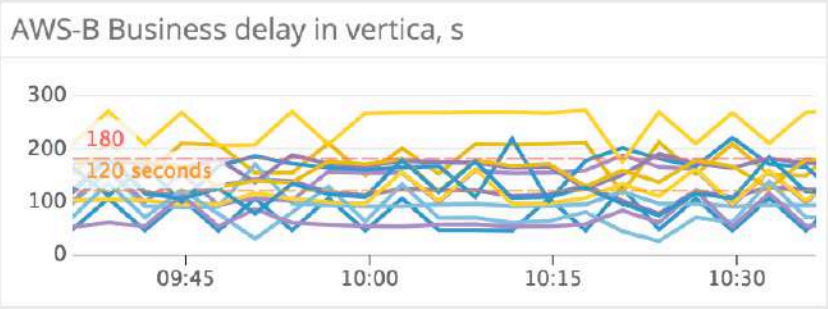


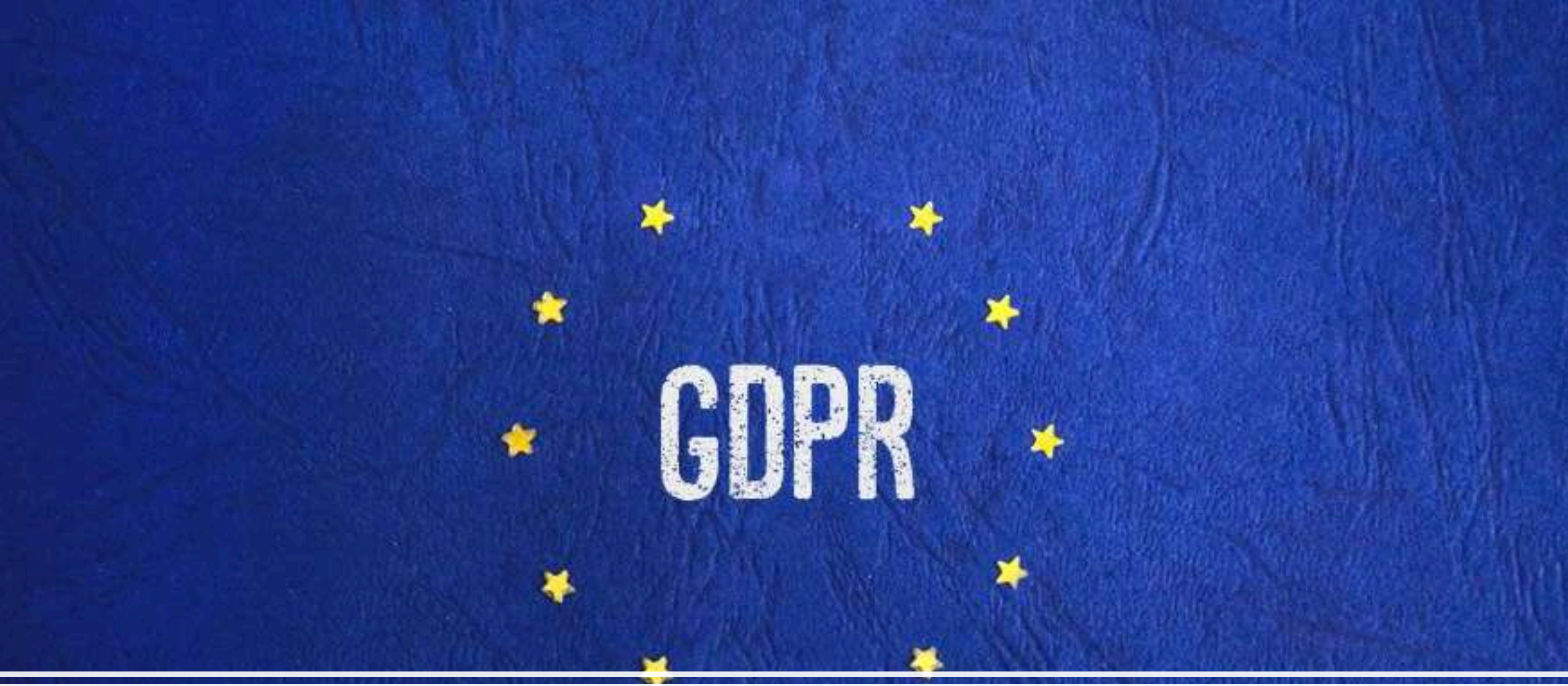






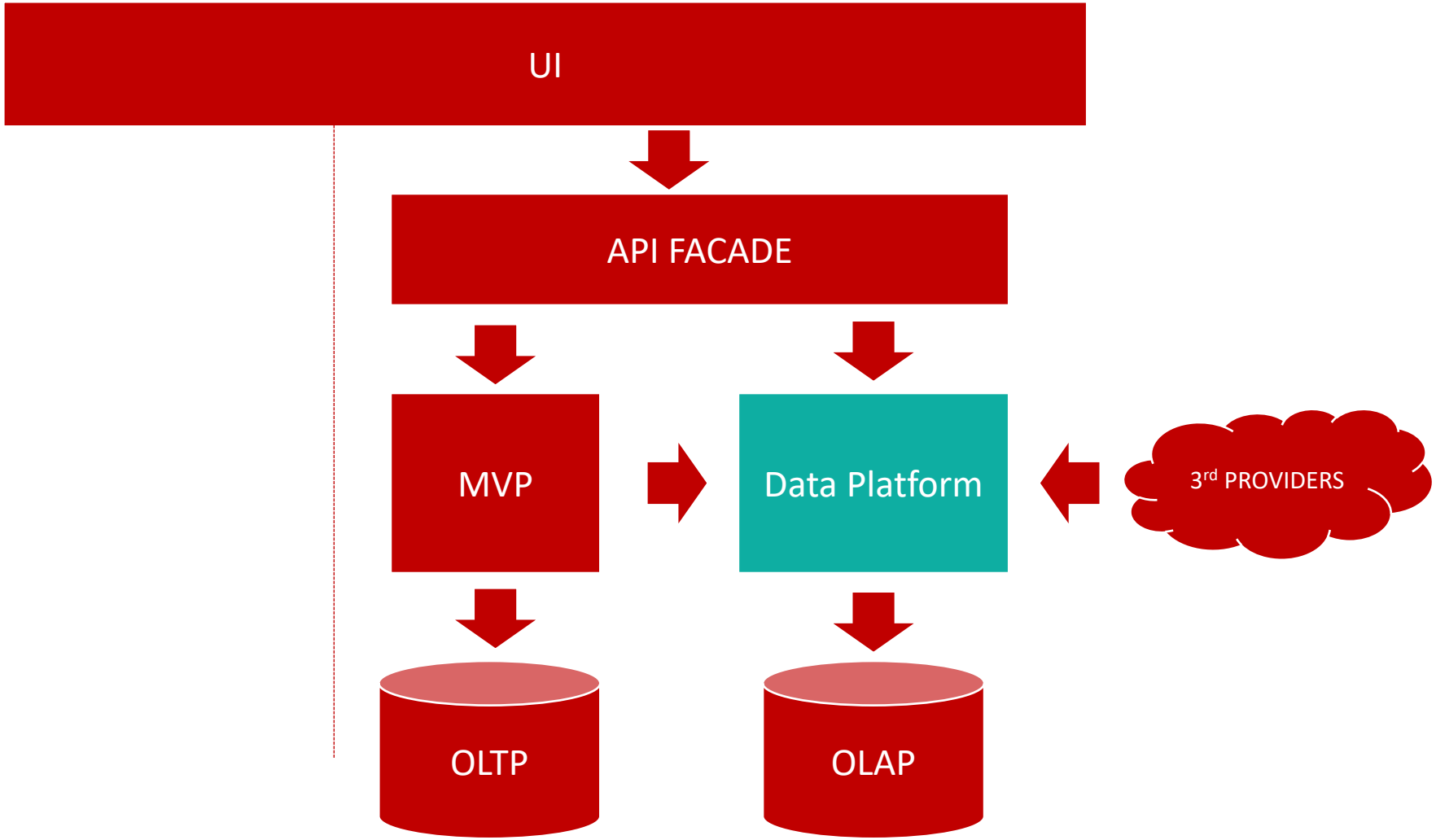




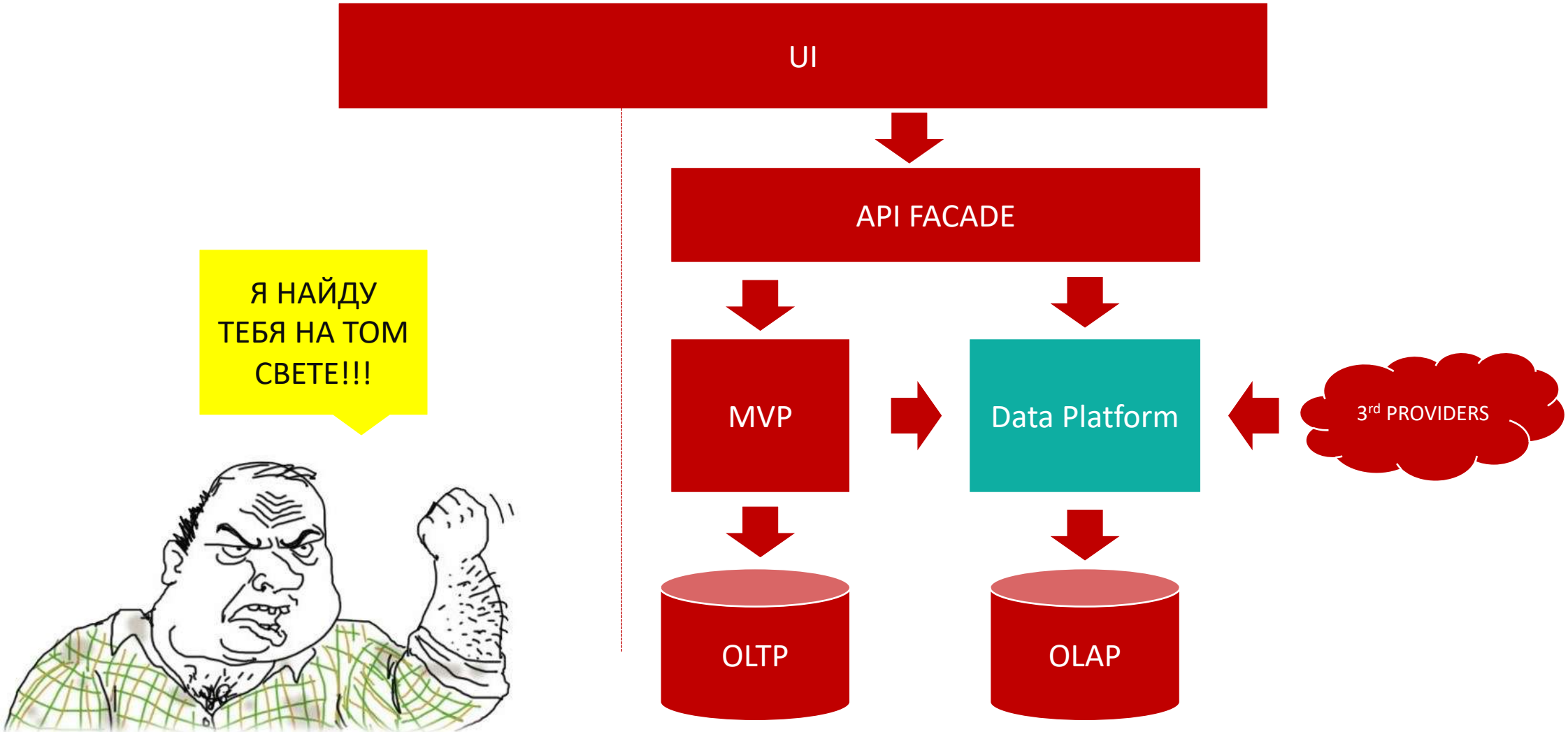


LESSON 6 – DATA GOVERNANCE

DATA GOVERNANCE



DATA GOVERNANCE



DATA GOVERNANCE CHECKLIST

- 😱 Did I think about Personal Data Protection?
- 😱 Did I think about Data Access Control?
- 😱 Did I think about Data Eviction?
- 😱 Did I think about Data Lineage?
- 😱 Did I think about Data Quality?
- 😱 Did I think about Data Inventory?

PERSONAL DATA PROTECTION

- Learn what Personally Identifiable Data (PID) is
- Think twice before storing any PID
- Anonymize data as soon as possible in ETL and prefer to use anonymized data over PID where never possible
- Introduce Anonymized Unique ID (AUID) and store relationship PID <-> AUID separately

DATA ACCESS CONTROL

- Introduce IAM for components and developers inside Data Lake and DWH
  Control access to PID and anonymized data

- Introduce ACL for end users inside OLAP
  Leverage OLAP features to support ACL -- per row, table, schema, database

DATA EVICTION

- Design data and applications with evection enabled
- Introduce data retention policy and schedule cleanup jobs
- Separate data retention policy per raw and aggregation tables
- Document retention policy

DATA LINEAGE

- Shit happens
- Shit will happen, think about it in advance

RECOMMENDATIONS

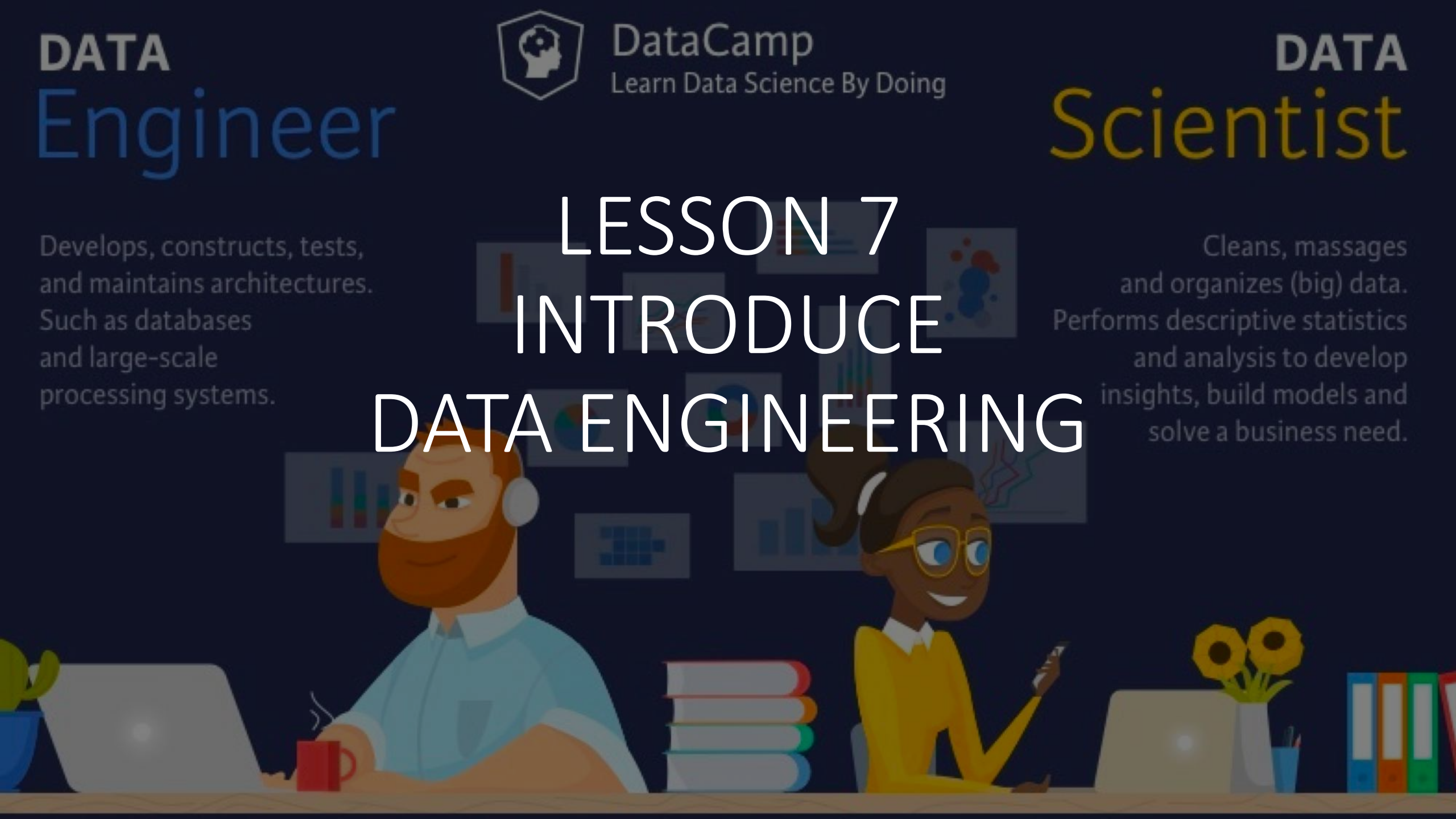
- Each ETL step should persist its output with reasonable retention policy
- Persist any application logs (Spark/Yarn, CMD apps, ETL, …)
- Log any significant application decisions
- Persist any provenance logs (NiFi, …)

DATA QUALITY

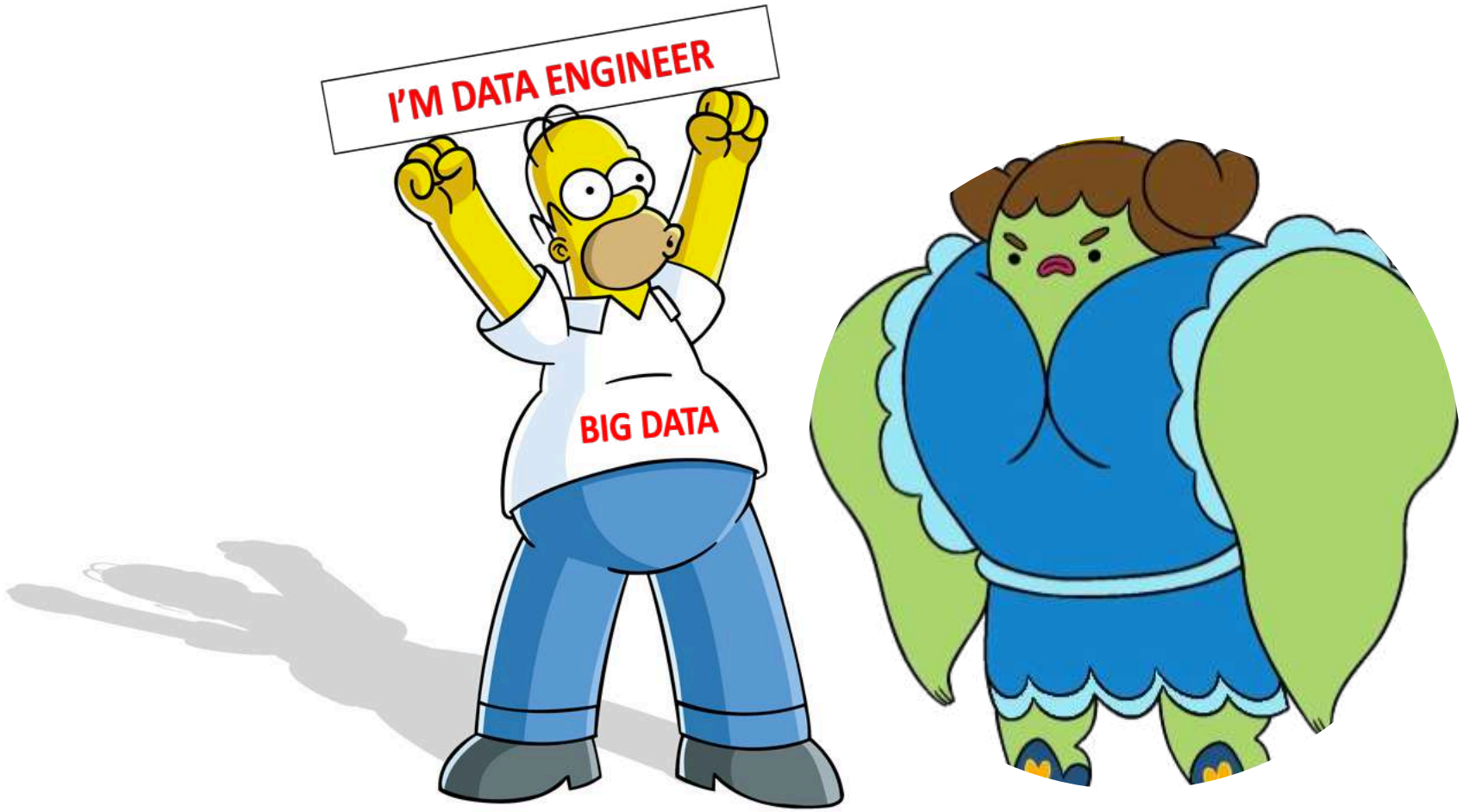
- Introduce data validation [even if it is undefined] and track validation issues
  - Schema errors (wrong type, missed mandatory field)
  - Semantic errors (unknown or poorly formatted IDs)
  - Business errors (certain business constraints per-event or cross-event)
- Track any errors and expose metrics
- Track discrepancies and expose metrics
  - raw and aggregation data
  - Discrepancy between real-time and batch
  - Discrepancy between vendor data

DATA INVENTORY

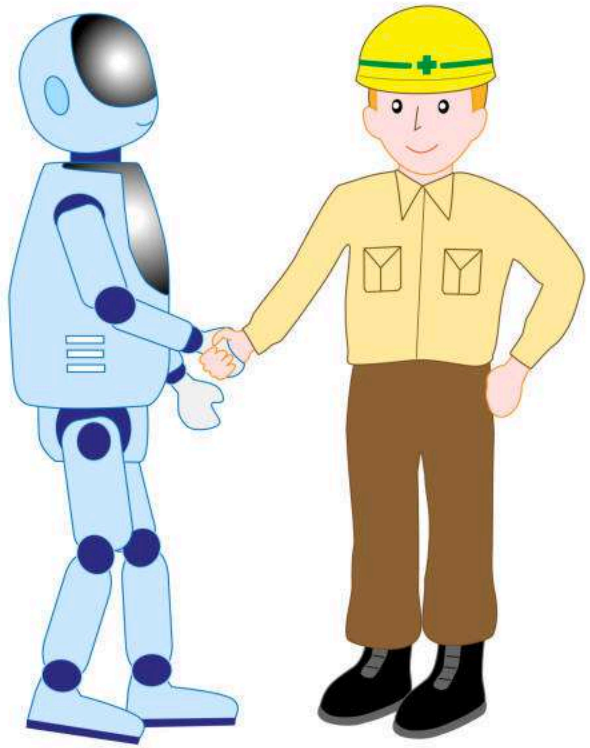
- Document how data organized
- Document where data stored
- Document what and where data exported
- Document what and where data ingested
- Document as granular as possible -- per vendor, data source, ETL component etc.



LESSON 7
INTRODUCE DATA ENGINEERING


I'M DATA ENGINEER
BIG DATA

SHARING RESPONCIBILITY TO DATA



Distinguish expertise

Involve Data Engineers to make Data Platform better and faster

THANK YOU