02. Data Pipeline

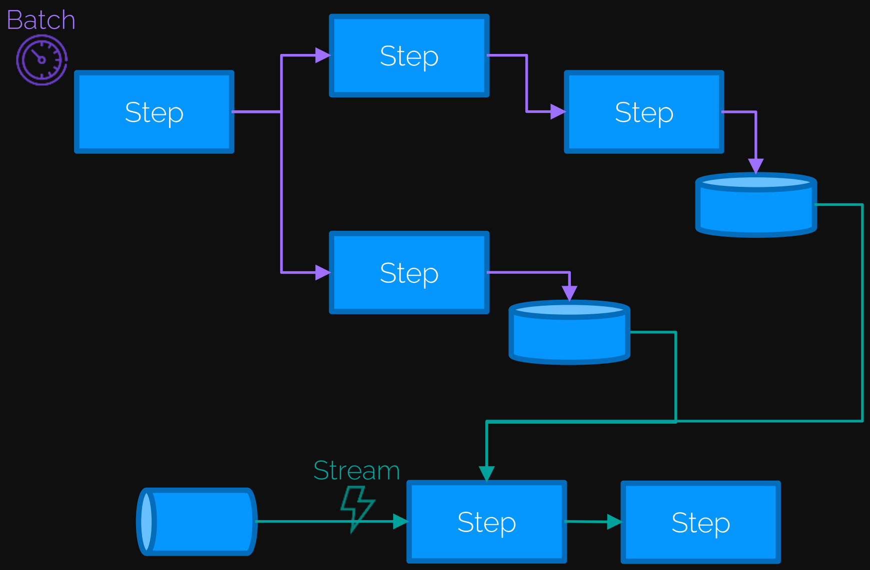
Transform data, Train & Evaluate model

What is a data pipeline ?
The basics

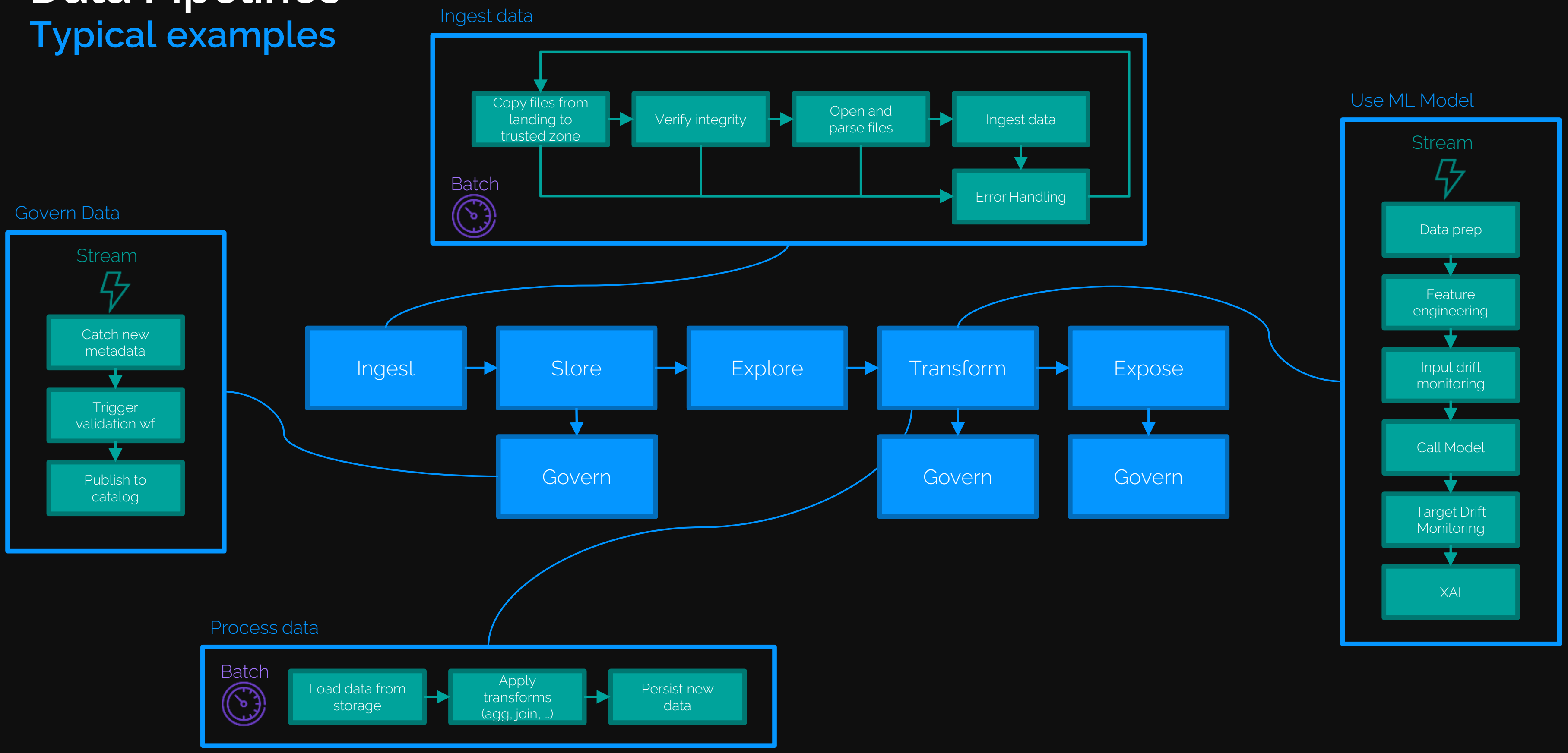
- Data pipeline = move data from A to B applying transformations
- Functionalities
  - Ingestion
  - Transformation (Filtering, masking, aggregations, cleansing, standardization, deduplication, ML models, …)
  - Storage

Technical Components

- Triggers
  - Batch
  - Stream
- Steps
  - Microservices approach : One business goal
  - Standardize interface (Rest API?)
- Chaining
  - Orchestration or choreography



Data Pipelines
Typical examples



Robust data pipelines

Things to keep in mind

- **Init mode**
  - Huge volumetry, optimize performance (chunks)
- **Failures handling**
  - Automatic retry
- **Retry mode**
  - Idempotency of every tasks
  - Avoid infinite loops
- **Observability**
  - Log management
  - Functionnal metrics
- **Reusability**
  - Variabilization, externalization of parameters (remember 12 factors)
  - Code once, reuse everywhere
- **Non-event**
  - How to handle lack of data
- **Scalability**
  - The weakest link in the chain will slow down everything (eg: panda processing with spark reading)

Triggers

When to launch a pipeline

How would you trigger a training pipeline ?

And a retrain pipeline ?

Time

Simpliest trigger available, based on clock
Standard cron format :

mi h md m wd

minute hour month_day month week_day

0 8 * * * : At 8:00 ever day
30 14 1 * * : At 14:30 the 1^{st} day of every month
00 23 * * 2 : At 23:00 every Tuesday

Event

Data

Streaming pipeline are listening for incoming data, they run every time they have new data

See next chapter for details

Notification

Messaging : a common way to trigger a stream or batch pipeline is to use an event notification providing all the necessary metadata for the run (ex: file trigger)

Webhook : other sort of notification, they are used

CDC

Change data capture detects when data are modified (CRUD) usually in strucutred data stores (Databases)

My rule of thumb : prefer complex pipeline with simple trigger rule

Atos

Software Craftsmanship

Maturity levels

123654851|2018-05-24|book|Tallinn, Estonia
123736540|2018-05-24|book|Doha, Qatar
123793204|2018-05-24|audio|Nashville, USA
123835264|2018-05-24|audio|Panama City, Panama
123862351|2018-05-24|book|Nabgkok, Tailand
123965841|2018-05-24|book|Bishkek, Kyrgyzstan

Pipeline

orderNb	date	item	itemNb	site	siteNb
123654851	2018-05-24	book	42	Tallinn, Estonia	412
123736540	2018-05-24	book	42	Doha, Qatar	155
123793204	2018-05-24	audio	15	Nashville, USA	632
123835264	2018-05-24	audio	15	Panama City, Panama	540
123862351	2018-05-24	book	42	Nabgkok, Tailand	325
123965841	2018-05-24	book	42	Bishkek, Kyrgyzstan	611

Lvl0 : Monolith

Read myfile.parquet
Normalize column item
Normalize column site

Append output to sales table

Lvl1: Specific ms

Read myfile.parquet

Normalize column item

Normalize column site

Append output to sales table

Lvl2: Generic ms

myfile.parquet

Read file $

item
itemNormalizerArray

Normalize column $

site
siteNormalizerArray

Normalize column $

sales

Append to table $

Lvl3: Idempotence (+ every best practices)

myfile.parquet

If file not already processed
Read file $

item
itemNormalizerArray

If not already normalized
Normalize column $

site
siteNormalizerArray

If not already normalized
Normalize column $

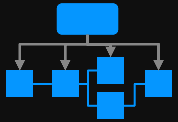
sales

Delete if already exists
Append to table $

Atos

Orchestration vs Choreography

Two different strategies

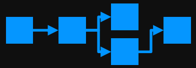


A master component is responsible to trigger every tasks of the pipeline, handle the results, combine them, retry if necessary, etc

Each musician in an orchestra master its own instrument, have its music sheet but collectivelly they're lost without the conductor

Centralized governance, easier monitoring

Central component (SPOF?), bottleneck
Not suited for streaming
Not good with huge amount of tasks/services



Every tasks of the pipeline is aware of where they get input information, what they have to do and where to send their status notifications

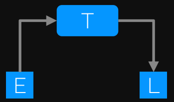
Dancers are listeting to the music and make necessary moves because they're all following the choregraphy

Alligned with microservice desing « dumb pipe, smart endpoints »

More complex services (they have to implement full logic)

ETL vs ELT

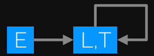
Make new out of old



Extract data from sources systems (operationnal db, IoT, CRM), needs great diversity of data connectors and triggers (eg CDC)
Transform data from a model/structure to another one, apply cleansing, agregate éléments, join with other sources, etc
Load phase is when resulted data is persisted on final storage. Sometimes, it can also be seen more widely with a sharing approach (cleaned data should be distributed to the rest of the enterprise)

Mature

Centrilized, monolith
Lowcode/nocode pipelines are hard to industrialize



Same stages than for ETL except than data is directly loaded into a storage solution design for analytics. Data transformation is then applied directly on this target storage usually using the query engine of this storage.

Raw data is available to business users
More accessible (SQL on lakehouse)
Better scalability and performance (today)

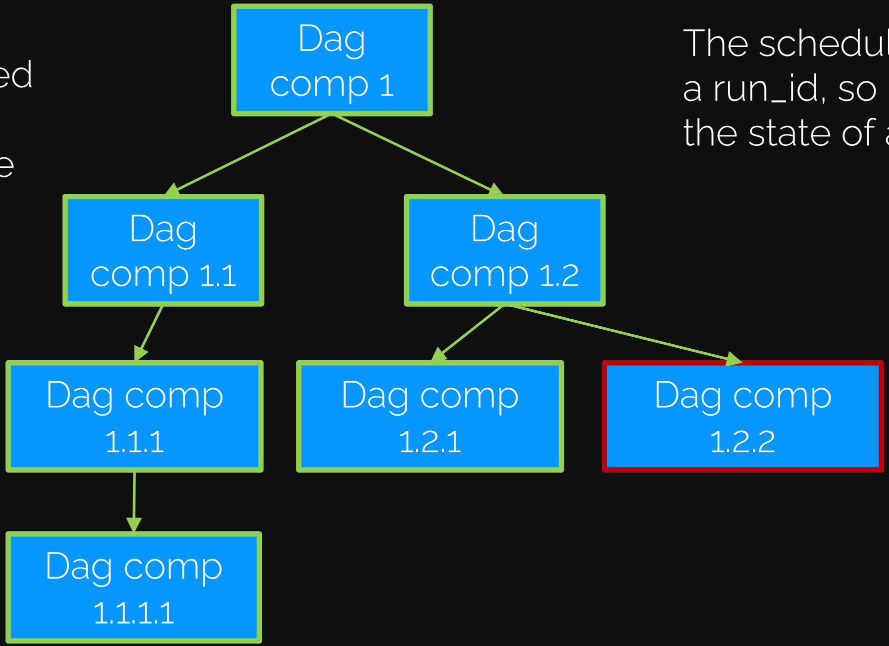
Warning with shadow IT !

+

Directed Acyclic Graph (DAG)

- DAG represents pipelines and make it executable with a scheduler

The DAG is defined by its state (composed by the state of its components).



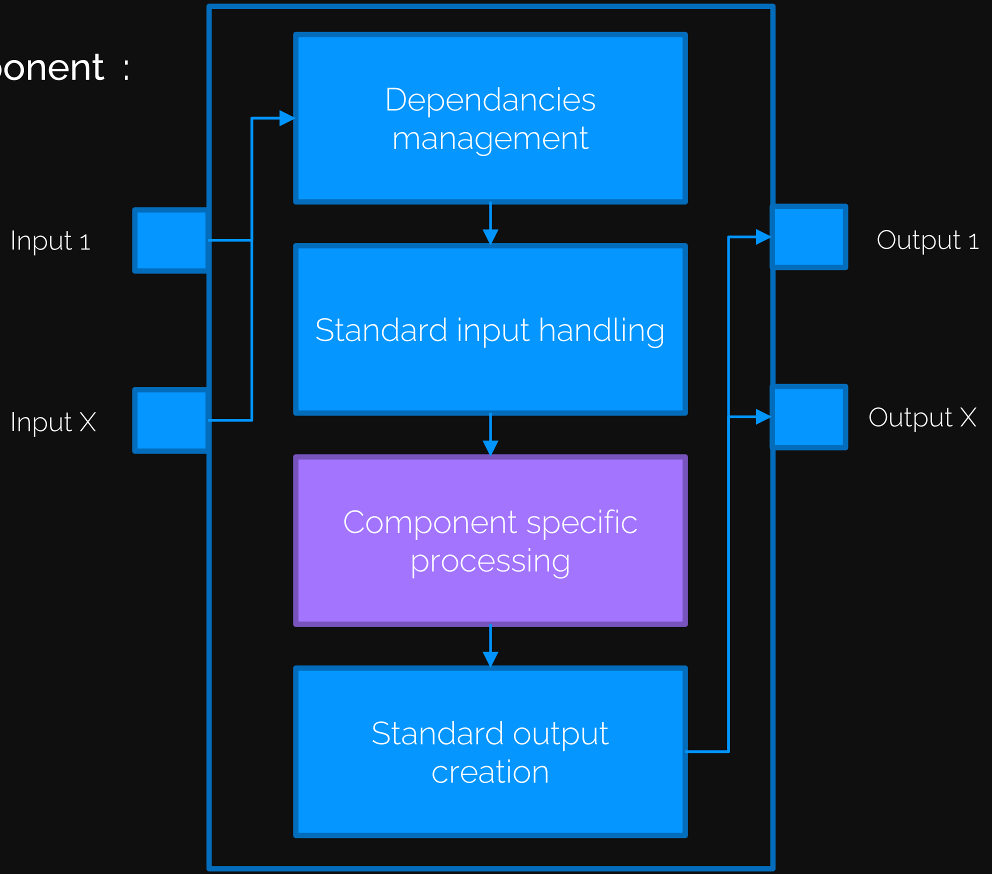
The scheduler assign a dag_id and a run_id, so it's possible to recover the state of a specific run_id.

Each component can sucess or fail, and the final state can dépend on this failure or not.

Templating

Reuse, reuse reuse !

- Component :



- Interface code responsible for pipeline integration, portability, as generic as possible

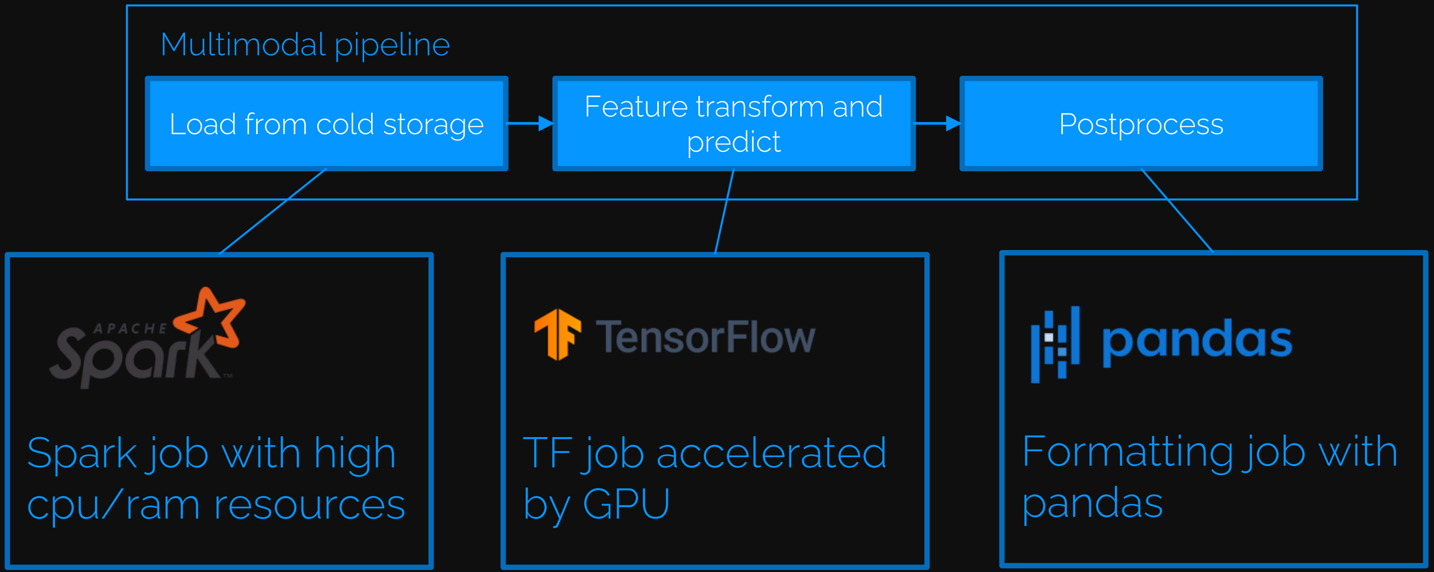
- Business code responsible for the component feature, specific

Atos

Heterogeneous processing

Feature transformation + model training + post processing

- Chaining components allow heterogeneous (code + execution) applications



Tracking experiments

Accelerate ML prototyping process with reporting

Params to track

Models level HP

- Topology

Training level HP

- Learning rate
- Regularization
- Optimizer
- NB epochs

Data/process level HP

- Dataset cut
- Label distribution
- Train set size

Results to track

Training process results

- Loss curve

Performance/ precision metrics

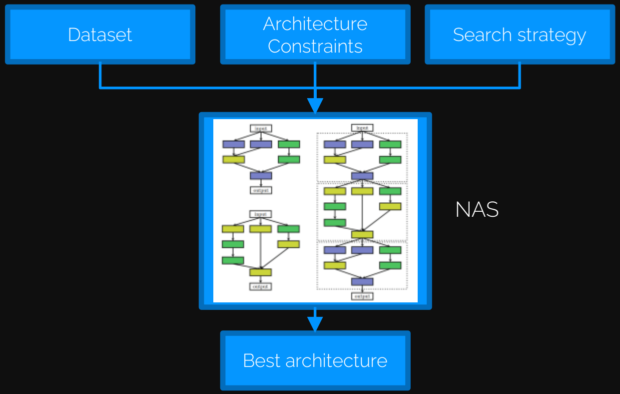
Aggregation And rendering

USECASE	Param1	Param2	Param3	Param4	Param5	ParamX	Accuracy	..
Run_MODELX	X1	X2	X3	X4	X5	Xx	0,76	
Run_MODELY	Y1	Y2	Y3	Y4	Y5	Yx	0,81	

AutoML
Optimized model selection

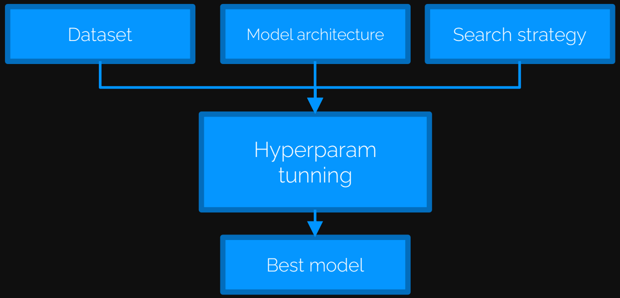
Neural Architecture Search

- Explore Model Architecture Space with search strategy for new candidates
  - Dimensions of the space : number of layers, type of connections, type of cells, …
  - Limiting space with hypothesis could help a lot, but introduction of biaises ?
- Train models and evaluate with Performance Estimation Strategy based on
  - model performance : metrics or low-fidelity proxy metrics (to accelerate eval)
  - architecture complexity : number of layers, overall size (number of weights), cells complexity
- It's computation intensive, so lots of research are made to reduce this task



Hyper parameters tunning

- Sub problem in AutoML systems
- Once the architecture is fixed, we can reach better model performance exploring the Hyper Parameter Space
  - Number of neurons, batch size, learning rate, etc



Quizz

What we've learn

Question				
Data pipelines are used to ingest and transform data	Y	N		
Common macro steps in data pipeline are design, build, run	Y	N		
Best practices when building robust data pipeline is to anticipate and handle various potential data failures	Y	N		
Idempotence is a best practice and brings parallelization in pipeline steps	Y	N		
With cron scheduling we can define irregular and dynamic intervalles	Y	N		
Which is better, orchestration or choreography ?	Orchestration	Choregraphy	Both	
Can we use event trigers to trigger pipeline step in orchestration mode (no choregraphy)	Y	N		
NAS and Hyperparameter tunning are part of AutoML	Y	N		

Quizz

What we've learn

Question				
Data pipelines are used to ingest and transform data	Y	N		
Common macro steps in data pipeline are design, build, run	Y	N		
Best practices when building robust data pipeline is to anticipate and handle various potential data failures	Y	N		
Idempotency is a best practice and brings parallelization in pipeline steps	Y	N		
With cron scheduling we can define irregular and dynamic intervalles	Y	N		
Which is better, orchestration or choreography ?	Orchestration	Choreography	Both	
Can we use event trigers to trigger pipeline step in orchestration mode (no choregraphy)	Y	N		
NAS and Hyperparameter tunning are part of AutoML	Y	N		

Common steps are ingest, store, transform and expose
Idempotency is the notion of rerunning with exact same effect
Orchestration is good for simple systems, choreography for complex ones, both are usefull

Atos

In Practice
Lab Content

- Batch processing+training
  - Exo1: Local pipeline
    - Train a custom model
    - Use tensorboard to follow training curves
  - Exo2: Simple KFP
    - Create a first component and a pipeline with it
    - Run the pipeline
    - Add custom metrics for the component and rerun pipeline
  - Exo3: Complex pipeline
    - Create components for essential steps (ingest, train, predict)
    - Assemble pipeline
    - Add more components