
MyScript


Seyos


CogniTalk


EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE


ML WEEK
THE MACHINE LEARNING WEEK

Event Sourcing applied to machine learning

Thomas Pocreau, @Talend

Do machine learning like the great engineer you are,
not like the great machine learning expert you aren’t.

Before Machine Learning

Rule #1: Don’t be afraid to launch a product without machine learning.
Machine learning is cool, but it requires data.

Rule #2: First, design and implement metrics.
It is better to get historical data now. Notice a problem? Add a metric to track it!

Rule #3: Choose machine learning over a complex heuristic.
A complex heuristic is unmaintainable. Once you have data and a basic idea of what you are trying to accomplish, move on to machine learning.

Context and Problem

- The mainstream approach is to maintain the current state of the data by updating it as users work with it.
- Typical software architectures are encouraging data lost.
- Views are built just before or after a create, read, update, and delete (CRUD) operation.

Machine learning is a unique opportunity to change the way we build software.

Terminology

- **Instance (Instance)**: The thing about which you want to make a prediction.
- **Label (Étiquette)**: An answer for a prediction task -- either the answer produced by a machine learning system, or the right answer supplied in training data.
- **Feature (Caractéristique)**: A property of an instance used in a prediction task.
- **Example (Exemple)**: An instance (with its features) and a label.
- **Model (Modèle)**: A representation of a prediction task. You train a model on examples then use the model to make predictions.
- **Metric (Statistique)**: A number that you care about. May or may not be directly optimized.
- **Objective (Objectif)**: A metric that your algorithm is trying to optimize.
- **Pipeline (Pipeline)**: The infrastructure surrounding a machine learning algorithm. Includes gathering the data from the front end, putting it into training data files, training one or more models, and exporting the models to production.

Your First Pipeline

Rule #4: Keep the first model simple and get the infrastructure right.

But you will run into many more infrastructure issues than you expect. Before anyone can use your fancy new machine learning system, you have to determine:

- How to get examples to your learning algorithm.
- A first cut as to what "good" and "bad" mean to your system.
- How to integrate your model into your application. You can either apply the model live, or pre-compute the model on examples offline and store the results in a table.

Your First Pipeline

Rule #4: Keep the first model simple and get the infrastructure right.

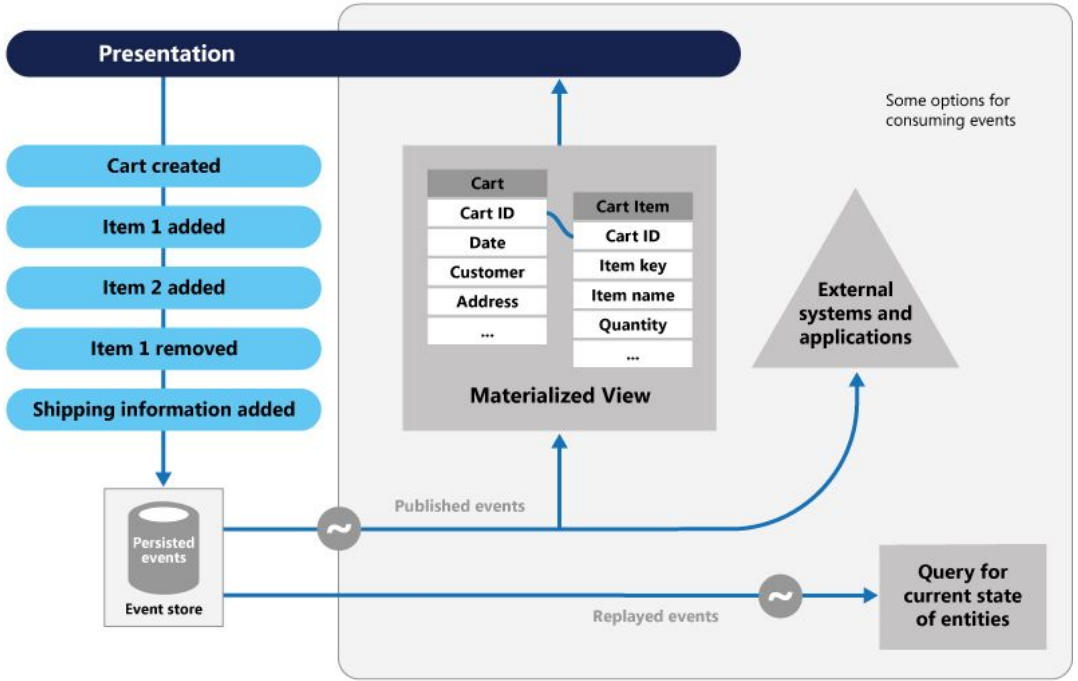
Choosing simple features makes it easier to ensure that:

- The features reach your learning algorithm correctly.
- The model learns reasonable weights.
- The features reach your model in the server correctly.

Once you have a system that does these three things reliably, you have done most of the work. Your simple model provides you with baseline metrics and a baseline behavior that you can use to test more complex models. Some teams aim for a "neutral" first launch: a first launch that explicitly de-prioritizes machine learning gains, to avoid getting distracted.

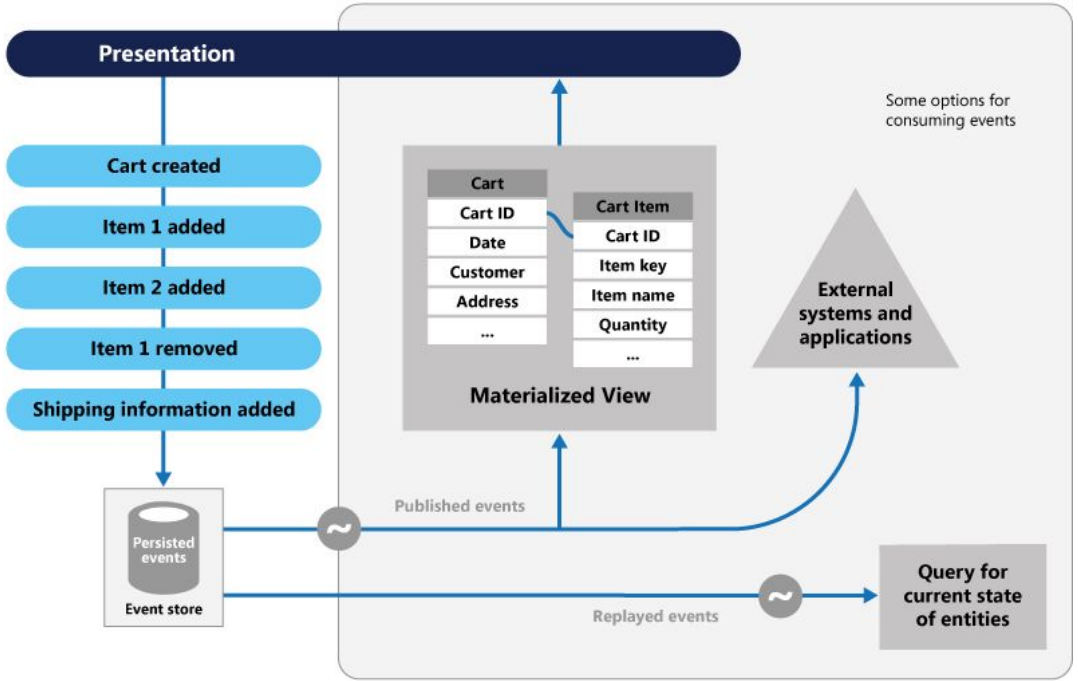
Event Sourcing pattern: Sequencing

The Event Sourcing pattern defines an approach to handling operations on data that's driven by a sequence of events, each of which is recorded in an append-only store.



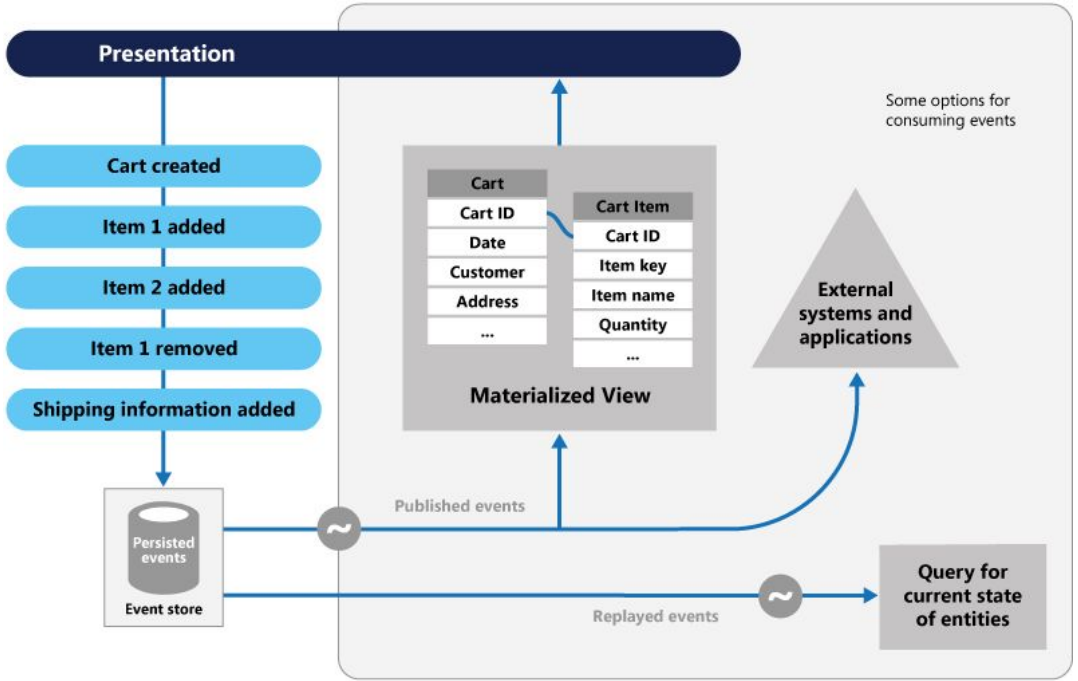
Event Sourcing pattern: Decoupling

Notice that the application code that generates the events is decoupled from the systems that subscribe to the events.



Event Sourcing pattern: Consuming

Typical uses of the events published by the event store are to maintain materialized views of entities as actions in the application change them, and for integration with external systems.



Event Sourcing pattern

The Event Sourcing pattern provides the following advantages:

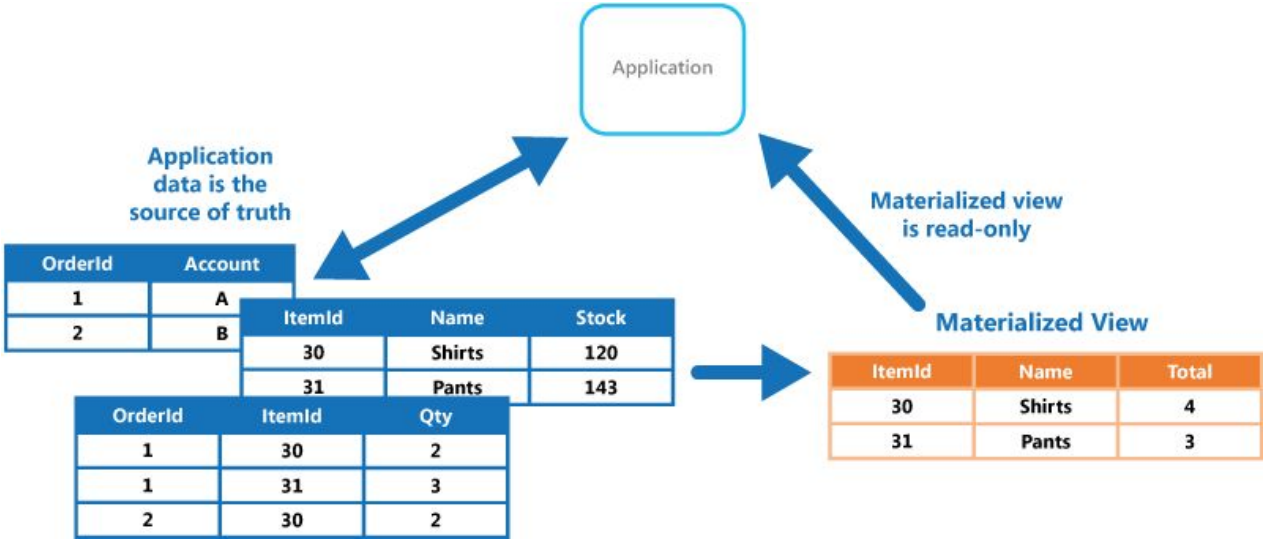
- Events are immutable and can be stored using an append-only operation.
- Events are simple objects that describe some action that occurred.
- Events typically have meaning for a domain expert.
- Ideal to capture intent, purpose, or reason in the data.
- you record events that occur, and you can to replay them.

Not useful in the following situations:

- Small or simple domains, systems that have little or no business logic
- Systems where consistency and real-time updates to the views of the data are required.
- Systems where audit trails, history, and capabilities to roll back and replay actions are not required.

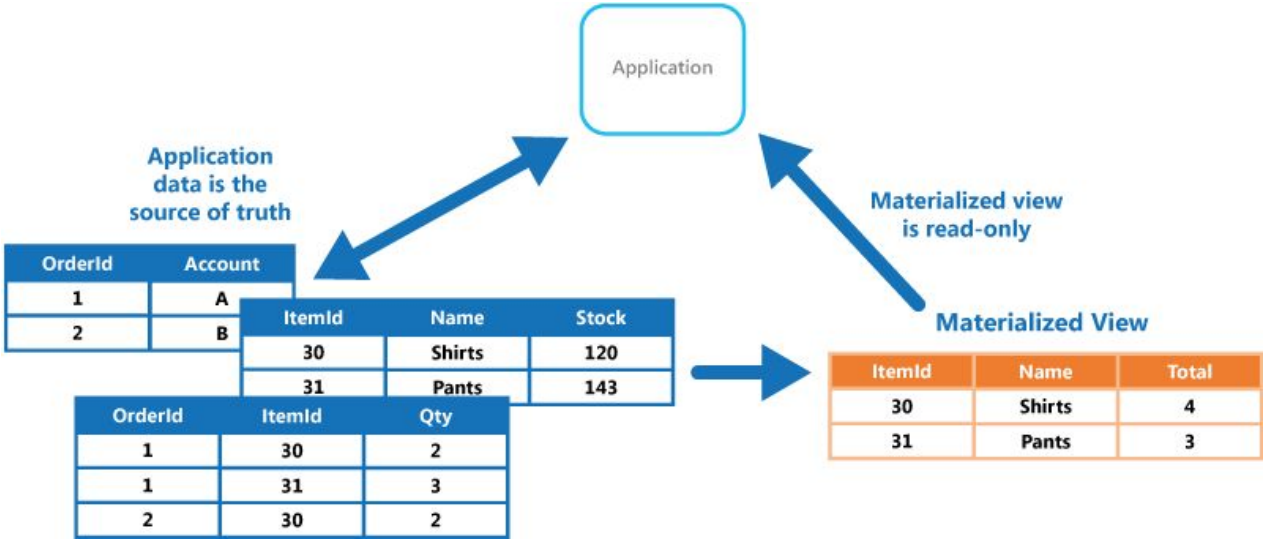
Materialized View pattern

To support efficient querying, a common solution is to generate, in advance, a view that materializes the data in a format suited to the required results set.



Materialized View pattern

The Materialized View pattern describes generating prepopulated views of data in environments where the source data isn't in a suitable format for querying, or where query performance is poor due to the nature of the data or the data store.

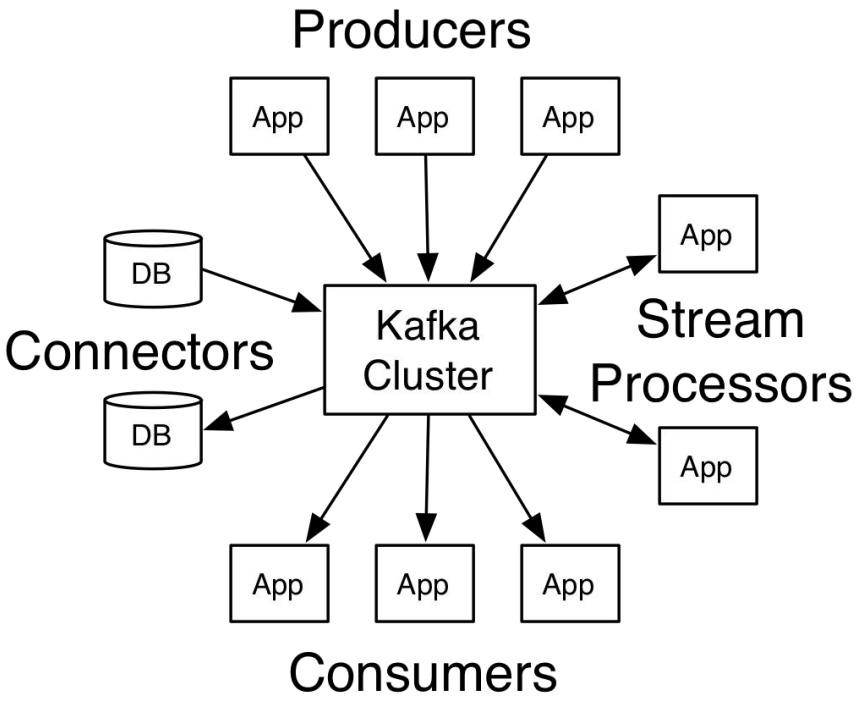


Materialized View pattern

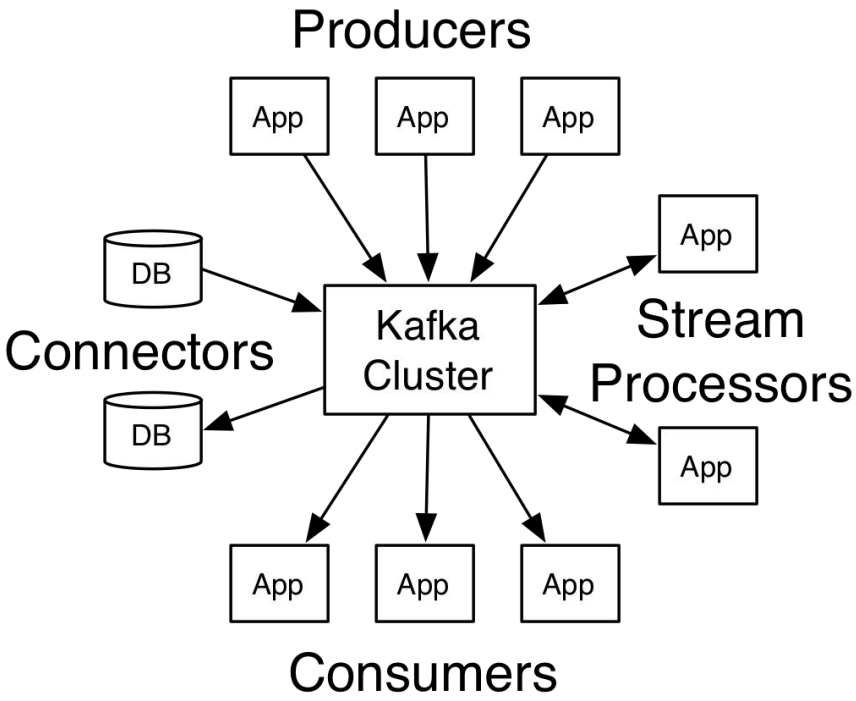
When to use this pattern:

- Creating materialized views over data that's difficult to query directly, or where queries must be very complex to extract data that's stored in a normalized, semi-structured, or unstructured way.
- Simplifying queries and exposing data for experimentation in a way that doesn't require knowledge of the source data format.
- Providing access to specific subsets of the source data that, for security or privacy reasons, shouldn't be generally accessible, open to modification, or fully exposed to users.
- Bridging different data stores, to take advantage of their individual capabilities.

Kafka to stream fresh events



S3/Minio to store cold events



Your First Pipeline

Rule #5: Test the infrastructure independently from the machine learning.

Make sure that the infrastructure is testable, and that the learning parts of the system are encapsulated so that you can test everything around it.

Rule #6: Be careful when copying pipelines.

Often we create a pipeline by copying an existing pipeline (i.e., cargo cult programming), and the old pipeline drops data that we need for the new pipeline.

Examples

Colisweb, iAdvize, VP, Talend

A model is a materialized view over prediction events.

Eat your own dog food

- **For each tensorflow prediction, store PredictionEvent(features,labels)**
- **Provide an api to tag PredictionEvent as an example in training dataset**
- **Train a model by building a materialized view over a subset of prediction events**
- **Validate your model by testing your newly trained model on past PredictionEvent**

ML Engineer ≠ DevOps

- **Use the same tools as the development team**
- **Don’t build pipeline outside the production environment**
- **Data lakes are for reporting, visualization, analytics, not machine learning**
- **Use the async apis provided by the team exposing the service, don’t maintain your own**
- **You should not have access to SQL databases of other micro-services**

Questions ?


MyScript


Seyos


CogniTalk


EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE


ML WEEK
THE MACHINE LEARNING WEEK