RAW Data Product Platform

White Paper

December 2021

RAW Labs SA
EPFL Innovation Park
Building I
1015 Lausanne
Switzerland
hello@raw-labs.com
https://raw-labs.com

Data challenges that RAW solves

Companies are struggling to keep up with the demands of their users for more data, from more sources and delivered in more ways. The number of data engineers is rising 40-50% year-on-year, and yet data delivery times are getting longer as environments get more complex with more tools, more platforms and more databases than ever as we move to a hybrid-cloud and multi-cloud world.

Other common problems cited include a lack of data engineering skills, engineers spending too much time on infrastructure, users not trusting the data and not being able to access data in the way they need it.

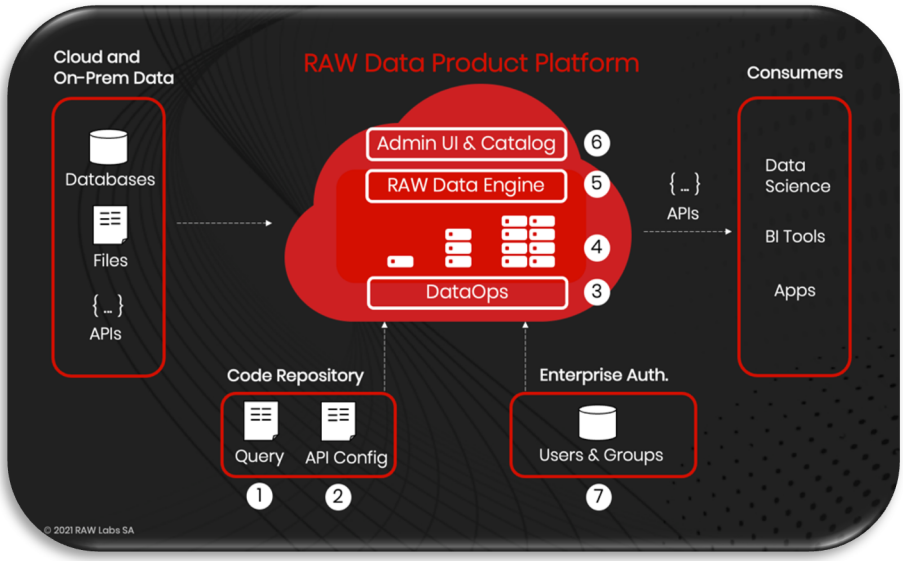
Centralising all the data into a single data lake or data warehouse does not scale, as bottlenecks inevitably occur on centralized data organizations - hence the rise of new architectures such as the "Data Mesh".

Data needs to be made accessible easily, via modern API mechanisms for system-to-system interoperability, and standard SQL interfaces for humans. It's clear that a new way of working is required and tools to support are needed.

RAW Data Product Platform

RAW is created with the mindset of both distributed data and distributed teams collaborating and delivering data products in a highly agile manner via low-code, DataOps approach. This can shorten the time and reduce the effort to create data products by as much as 10x.

RAW enables data accessibility by creating, publishing and managing data products as APIs and SQL interfaces which access data from databases, files and APIs located anywhere. RAW adopts a Platform-as-a-Service model, runs in Cloud environments and is scalable and secure for enterprise needs.



Company background

RAW Labs SA is a Swiss-based technology company with deep roots in academic database research. Founded in 2015 by Professor Anastasia Ailamaki and Miguel Branco, the company has taken its years of database research on scientific data and applied this to the business domain problem of creating data products. RAW is funded by serial technology investors with experience at Salesforce, Oracle, NTT and Credit Suisse.

Platform components

Key components of the RAW solution include:

① Low-code data query

RAW uses a dialect of standard SQL with simple functional language constructs to enable data access, query and manipulation inside a single powerful and easy-to-use tool. Query nested files (JSON, XML, etc.) or log files using the same simple constructs as tables, columns and flat files such as CSV. Web APIs are also directly accessible so that glue-code is not required.

② Configuration driven APIs

APIs are created by a simple YAML file, where the API's definition, metadata, security, and compute class are specified. The compute class can be set per API endpoint, allowing certain APIs needing more resources to use a large cluster. RAW supports multiple resource groups.

③ DataOps environment

RAW harnesses the power of your collaborative software code management environment (e.g. Git) so that data delivery can use the same tools as software delivery, using both DataOps and Data as Code approaches means that iterations are much faster and produce better quality outputs, and everything is tracked and versioned accordingly. RAW syncs with Git automatically so that APIs are immediately available including branches for testing purposes.

④ Platform-as-a-Service

The RAW platform is fully serverless and runs inside Cloud-based environments where multiple resource groups are made available for queries. Single and Multiple tenant options are available including the option to host inside an Enterprise's Cloud account. Billing is highly granular, as all API executions utilize credits that depend on the resources and queries executed on resource groups. The queries are fully isolated from each other using containers, and all data can be encrypted both in-motion and at-rest.

⑤ RAW Data Engine

The RAW engine is the result of years of research and enables scale-out query execution for distributed, heterogenous data and complex data types. Based on unique monoid comprehension calculus, the engine is built with state-of-the-art 'JIT' code generation techniques. This creates high-performance code which also uses a smart caching algorithm with multiple types of cache employed.

⑥ Administration UI and API Catalog

A UI to administer and view the catalog of APIs, connect to Git repositories, as well as review activity performance, spend, monitor logs and maintain users and RAW roles. APIs can be exported via OpenAPI and integrated with API management tools and catalogs.

⑦ Enterprise security integration

RAW integrates with Enterprise security systems via standards-based OAuth (Auth0) mechanism, supporting security policies and scopes for controlling data and API entitlements. API Access is fully monitored and can integrate with your API management platform.

Example of how to use RAW

1. Set up RAW to connect to your Git repository. This is a one-time activity. RAW will be given access to read from the repository where the query and configuration reside.

2. Create a query function that returns data:

```
movies := READ("https://raw.githubusercontent.com/prust/wikipedia-movie-data/master/movies.json");

search_by_title(title: string) := {
SELECT * FROM movies m
WHERE LOWER(m.title) LIKE
"%" + LOWER(title) + "%"
}
```

   And save as "movies.rql" in a path of "/1/public/movies"

3. Configure the query to create an API endpoint:

```
raw: 1.0
endpoint: GET
metadata:
  title: Movies by title
  description: Get movies by title
  tags:
    - movies
    - wikipedia
code: rql
codeFile: movies.rql
declaration: search_by_title
format: json
security:
  scopes:
    -  sales
    -  marketing
computeClass: standard
enabled: true
```

   And save as "movies-by-title.yml" in the same directory

4. Commit to Git in the usual manner and start using:

```
https://demos.raw-labs.com/api/1/public/movies/movies-by-title?title=jaws
```

   This example assumes the user has access to the endpoint via the scopes, which are Auth0 constructs for entitlements. There are also public endpoints.

