Safe Repository

Overview

"Safe Repository" will be a solution to store streams of experimental data for the Hadron Collider at CERN and make them available to scientists. The data will be delivered to the solution in the form of messages from a Message Broker. Data will be made available through a REST interface. There will be three resources: users, experiments, and measurements.

Users

Users will represent the users authorized to interact with the solution. There will be two user types: administrators and scientists with the following role matrix:

Administrators

resource	scope	access
users	complete	RW
experiments	complete	RW
measurements	complete	R

Scientists

resource	scope	access
users	user's record	RW
experiments	only records associated with user	RW

measurements	only records associated with user's experiments	R

Experiments

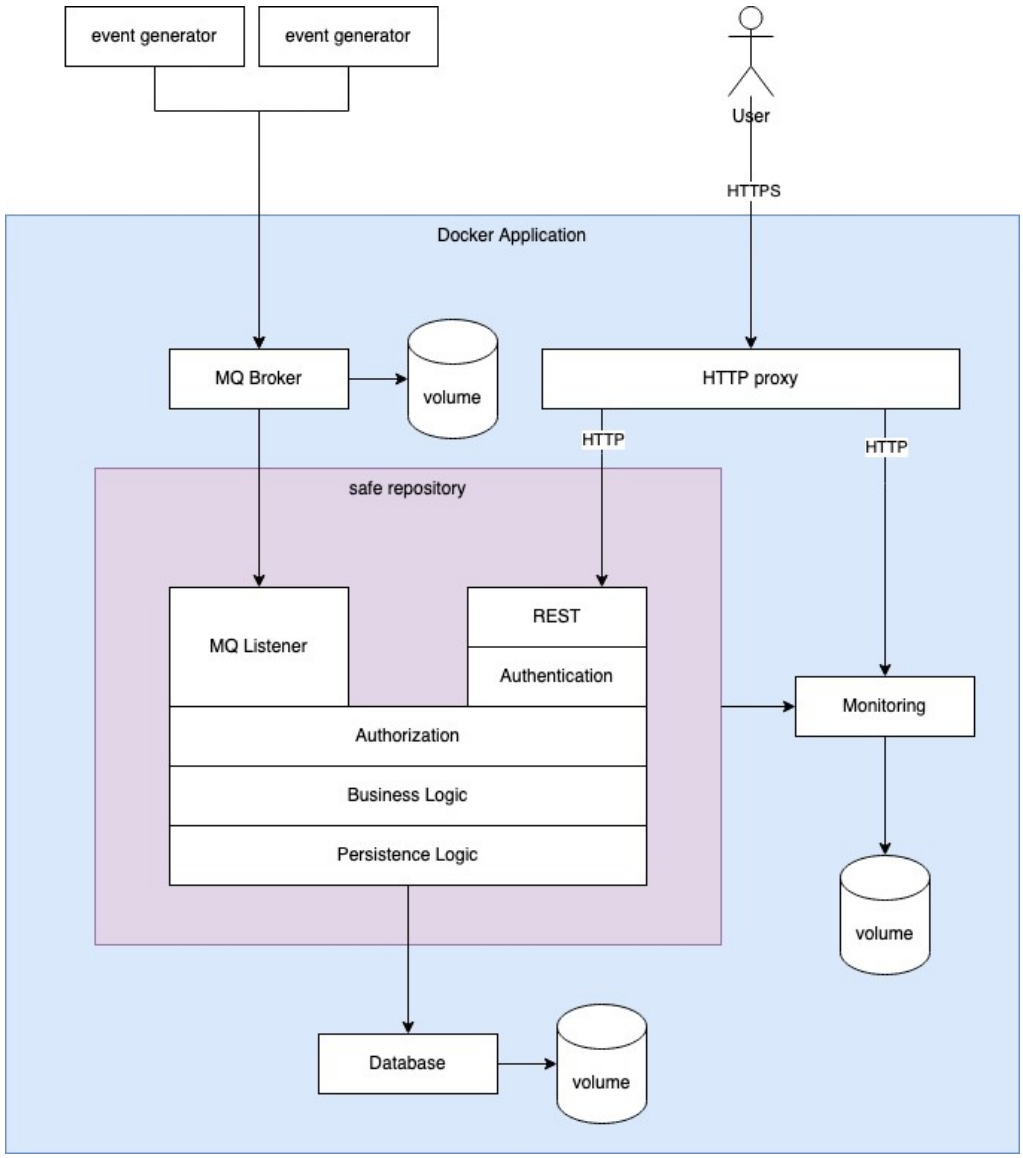
Experiments will represent single experiments. Only administrators and scientists associated with the experiment will have access to an experiment.

Data

Raw data from the measurements. It will be a read-only resource associated with an experiment displaying the measurements collected from the Message Broker.

Components

The solution comprises multiple components responsible for different aspects. The solution (in the blue area of the diagram) will interact with external data producers and users.



Deployment

The entire solution will be containerized with Docker and orchestrated with Docker Compose. This choice makes it easier to deploy the solution in other platforms such

as Kubernetes with minimal changes to the configuration. This strategy also aims to reduce the attack surface by hiding all the services that do not need to be reachable by the user.

HTTP Input

An HTTP proxy is responsible for encrypting the communications with the solution, offloading the incoming encryption, and adding encryption to the output. All HTTP-based communication will flow through the proxy hiding the corresponding services.

For this purpose, the implementation will use Nginx, which could also implement load-balancing functions should it be necessary to scale the application with additional nodes.

Data Streams

An MQ Broker will expose queues to accept data streams from the experiments.

RabbitMQ is one of the most popular tools on the market for this particular need. RabbiMQ supports TLS to encrypt communications and clustering to address scalability issues.

Storage

A database is responsible for the storage of the application's data. Being the model of the data known, a SQL Database represents a simple solution. Further expansions could consider moving the experiments' data to a NoSQL database that may offer better scalability.

There are no special requirements to guide the choice of a product. MySQL will be the initial choice. The application's design will ensure a level of abstraction to minimize the impact of a change should new insights suggest a different product would be a better fit.

Safe Repository

Safe Repository is the core component responsible for storing Hadron Collider's experimental data. It will expose multiple HTTP endpoints to configure the application resources and will process parallel data streams through input queues. The application stores the information on an external database. Safe Repository will not be exposed directly to outside traffic.

Safe Repository will be implemented in Python using Flask to expose REST APIs and Pika to process MQ Queues.

Filesystem encryption

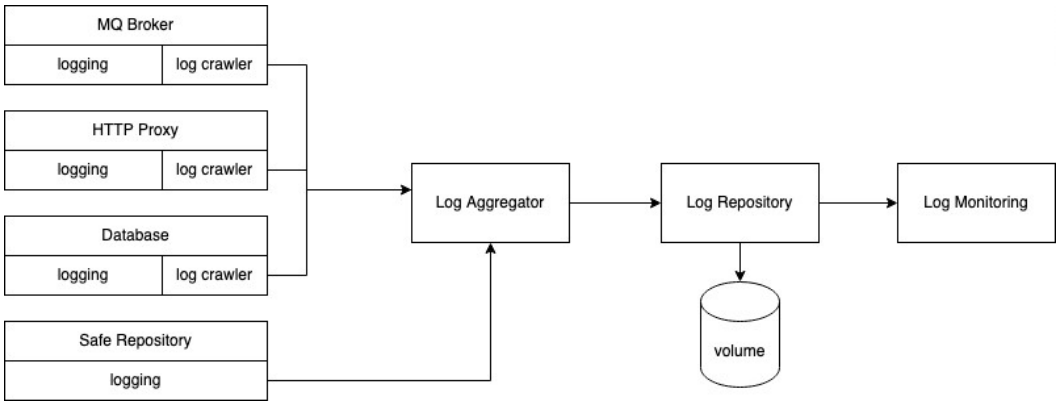
Filesystem encryption can be used to host the filesystems of the storage and the broker. This configuration is transparent for the solution.

Logging and Monitoring

All services will forward their logs to a centralized log aggregator responsible for the parsing and storage into a database. The logs will be visible in a dashboard exposed through the HTTP proxy.

This requirement will be implemented with the ELK Stack. Except for Safe Repository, all services will include Filebeats to forward their logs to Logstash. Safe

Repository will send the logs directly to Logstash instead. Logstash will be responsible for the parsing and the storage in Elasticsearch. Kibana will expose a dashboard to let users monitor the functioning of the whole system.



Dimensioning the system

Disk

User and experiment data will require less than 1Kb per record, and their number is expected to be in the range of thousands. Therefore it is safe to assume that a few megabytes will be sufficient to store them.

Each measurement is expected to require at least 22 bytes.

- 2 bytes per measurement type
- 8 bytes per timestamp
- 4 bytes per experiment id
- 8 bytes per measure

With 1 million measures per experiment, each experiment will require about 21Mb of space.

CPU and memory

TBD

Scalability

Periodic pressure on resources

Except for the database, all components of the solution will be stateless and allow for horizontal scalability. Clustering solutions such as Kubernetes have autoscale functionality to regulate the number of running instances and deal with variable demand. Autoscale functions are also efficient to contain the costs since additional CPUs and memory are allocated and billed only when required.

SQL databases scale well vertically, and it will be easier to scale the number of resources.

Interactive response requirements between request and reply

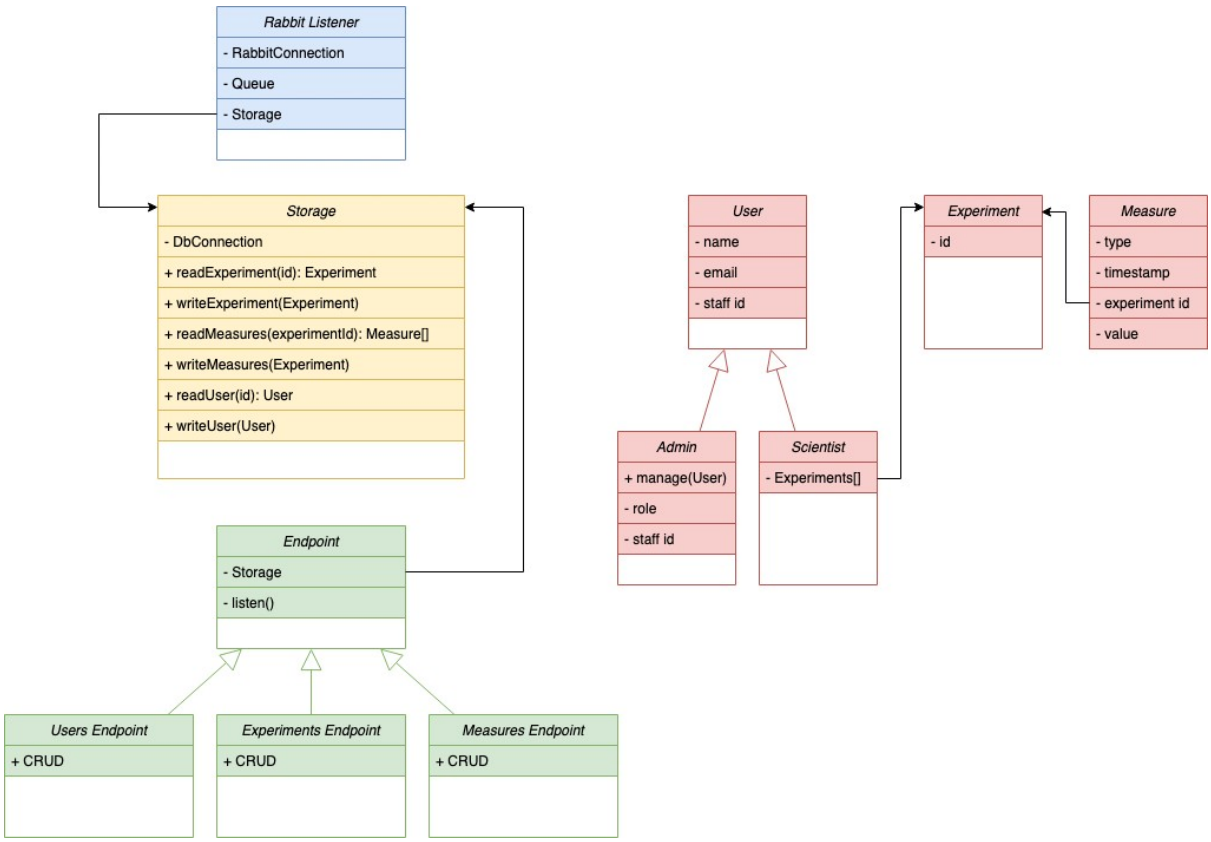
TBD

Substantial data download requirements

It is expected an elevated flow of data coming from queues while users will visualize them almost in real-time. The database will act as a buffer between the component responsible for storing the incoming data and the component responsible to make it available to the users. It is expected that the highest pressure could come from the input flow. Since reading a large amount of data could put pressure on the database, queries should be paginated.

Application Layer

The following diagram illustrates layers of the application and the interaction of the components.



There will be no direct interactions between the components consuming messages from the broker (in blue) and the components exposing REST endpoints (in green). The Storage (in yellow) will mediate the communications between the two parts.

Authentication

An authentication endpoint will validate credentials comparing the input with the hashes stored in the database. The endpoint will return a JSON web token that will remain valid for a limited time. The token will be required in the calls to all other APIs. The authentication endpoint will require the inclusion of a shared secret (API key) in the request. This additional measure will limit the chances to perform a brute force attack.

Authorization

Authorization will be handled at the level of the storage. Whenever a user will perform a query, his identity will be part of the query.

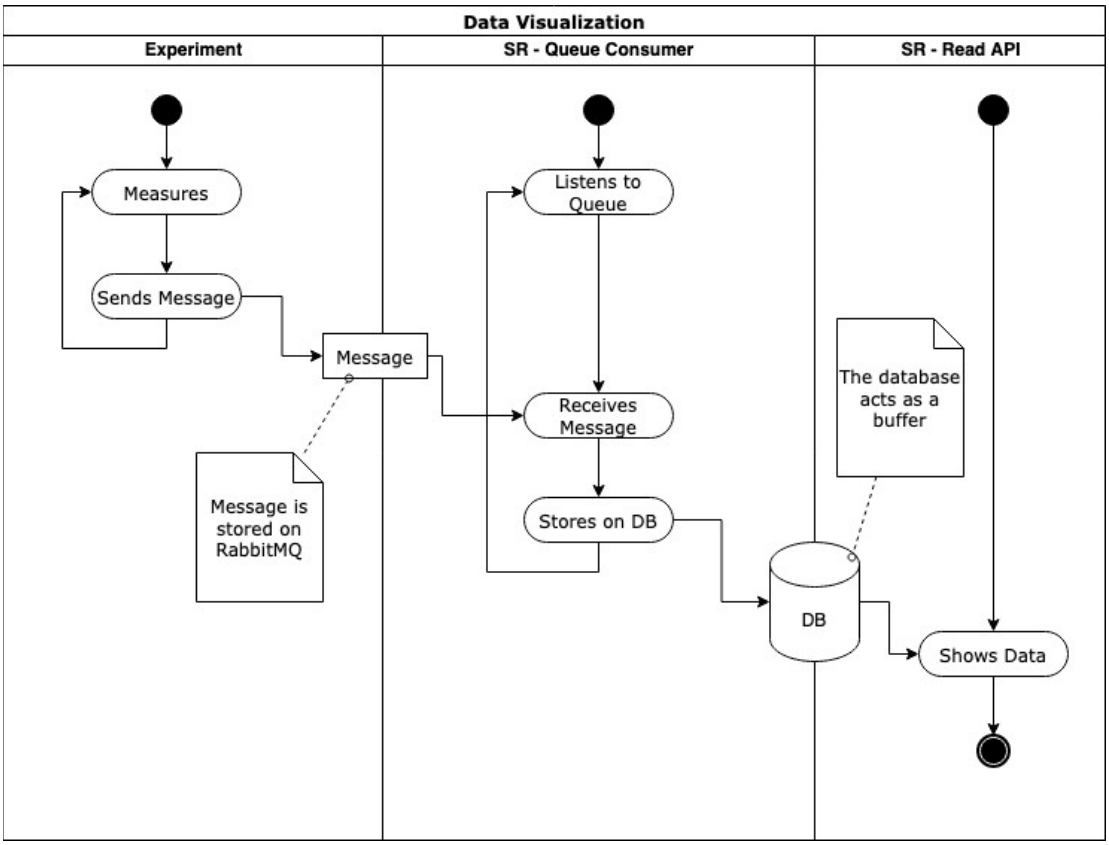
For example:

```
SELECT d.* from user_experiments ue, data d
where d.experiment_id = :experiment_id -- clause to select the data
and ue.experiment_id = d.experiment_id -- join
and ue.user_id = :user_id              -- safety measure
```

If the id of the current user is not associated with the experiment, the result of the query will be empty thanks to the last two lines.

Usage

The following diagrams illustrate the flow of information in the application.

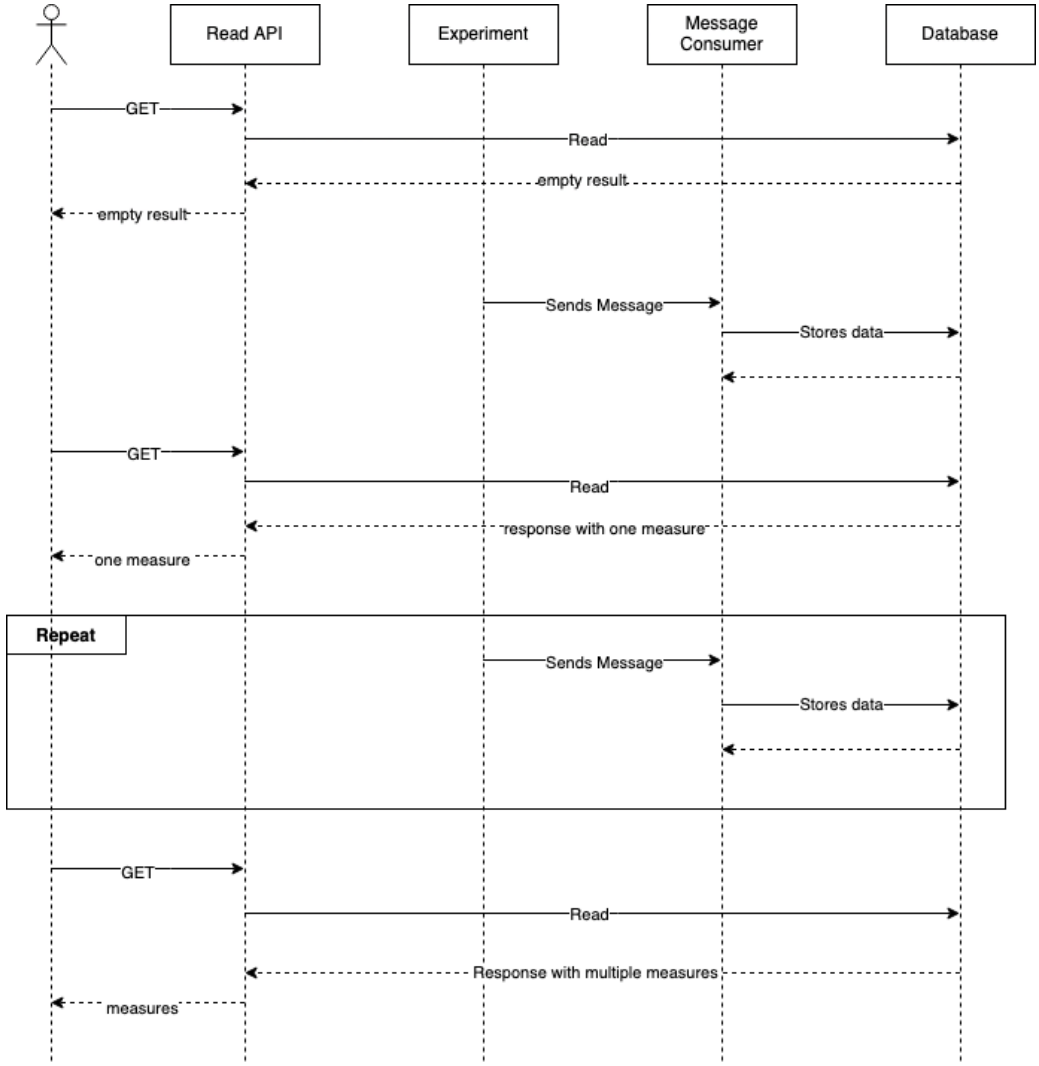


The component performing measurements (out of scope) will send measures to the application's input queue.

A thread in Safe Repository will be responsible for reading the messages and storing their content on the Database.

Safe Repository's APIs will display the content of the database on demand.

From a timeline perspective, repeated calls to the APIs will return more results as more data is inserted into the database.


Read API
Experiment
Message Consumer
Database
GET
Read
empty result
empty result
Sends Message
Stores data
GET
Read
response with one measure
one measure
Repeat
Sends Message
Stores data
GET
Read
Response with multiple measures
measures

Security

Using the STRIDE model, the following threats were identified and classified with DREAD.

Spoofing

User's credentials violation

Type	Level
Damage	High, experiments would be exposed, users' records compromised, data leak
Reproducibility	High
Exploitability	High
Affected users	One user. All, if the user is administrator
Discoverability	Medium. User's credentials may be easy to guess

Tampering

Introducing fake measurements on the message broker

Type	Level
Damage	High, experiments would be invalidated
Reproducibility	Medium. The highest risk is broker's authentication
Exploitability	High. Discovering credentials would make it easy to exploit the vulnerability
Affected users	All scientists
Discoverability	Medium. The broker is public, but credentials are highly secure

Repudiation

Scientists will not be able to manipulate the association between them and the experiments, or between the experiments and the data. Data will not be editable.

Information disclosure

Database breach

Type	Level
Damage	High, data would be exposed
Reproducibility	Low. Database is not directly exposed, authentication is in place
Exploitability	Low. Attacker should compromise at least another system first
Affected users	All
Discoverability	Low

Denial of service

DDos on APIs

Type	Level
Damage	High, system may become inoperative
Reproducibility	Low. The system should be exposed only in the internal network
Exploitability	Low. It would be easy to block the attack in the internal network
Affected users	All
Discoverability	Low. It would be difficult to plan an effective attack.

Elevation of privilege

Scientists becoming administrators

Type	Level
Damage	High, the attacker could disrupt the system
Reproducibility	Low. It would require database access since no system function manipulates roles
Exploitability	Low. Attacker should compromise at least another system first
Affected users	All
Discoverability	Low

GDPR Consideration

The application design requires only a minimal amount of personal information. Such information includes a staff identification number, name and surname, and email address. All users will be able to retrieve, update and delete their own information, in compliance with GDPR. For users unable to access the application, an administrator will be responsible for any GDPR request.