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.

Tools

- Elastic Search - Elasticsearch is highly scalable, it will allow to store, search, and analyze huge volumes of data in near real-time. It is able to achieve fast search responses as Elasticsearch searches an index instead of text directly.

- Kibana – Will allow to visualize data and monitor the application. It offers a powerful and easy-to-use visualization tool such as line graphs, pie charts, heat maps, and histograms. Kibana sits on top of the Elasticsearch stack in turn provide data visualization capabilities for data indexed in Elasticsearch.
- RabbitMQ - Is an open-source message broker application. Which will accept messages from vendor and delivers them to consumers. RabbitMQ acts as a middleman. It will reduce delivery times usually taken by web application servers.
- MySQL - MySQL will allow to handle, store, modify and delete data and store data.
- Nginx – Is an open-source software used for web serving, reverse proxying, caching, load balancing, media streaming, and etc. But for this project Nginx is used as a load balancer which will distribute application traffic across a number of servers.
- Firebase Authentication – Firebase is an extensible token-based auth system which supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and etc. Firebase Authentication leverages industry standards such as OAuth 2.0 and OpenID Connect.
- Logstash – Is an open-source server-side data processing pipeline that allows you to collect data from Elasticsearch.
- Filebeat - Is a lightweight file harvester used to fetch log files from Logstash and centralizing said log data.
- Docker/docker compose - Docker will be used to containerize of the application to help with easy deployments and distribution.
- Python/flask - Flask is a Python web framework that will be used to create web applications in Python.

Components

The solution comprises multiple components responsible for different aspects.

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. (TBD encryption)

RabbitMQ is one of the most popular tools on the market for this particular need. (TBD usage of exchange and routing keys)

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.

(TBD encryption)

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.

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.

APIs

Once the API is called by the client, the solution will validate and authorize the client and return information based on the level of access they have and information they required. The response will be in Json format by default.

There will be only one web service base URL be exposed and that particular url will have various paths and methods for clients to decide and call based on the data it require.

Web Service:

When the GUI client requires data from the service, the client will establish a connection to the Web API authorization service (Yet to be Implemented) to generate a jwt token and passing the user credentials. The Web API will check the credentials and will issue “Access token” and “Refresh token” in the response as JSON value. These access token will be then used in the subsequent API method calls from the client. The Access token will have issue time and Expire time (expiry yet to be determined, usually 5 minutes). Once the token is about to be expired, the client has to provide the refresh token and get a new token. A user can Login from different IP address at the same time and will be issued different tokens for each login.

For audit purposes, the Web API will maintain an audit trail table for every request by logging user details, date & time, IP address, request and their downloaded record count.

The Web API will validate each user call against the user profile.

Scalability

Periodic pressure on resources

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.

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. For this reason (TBD)

Application Layer

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

(TBD)

Usage

The following diagram illustrates the flow of information in the application.

(TBD)

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

N.A.

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.