

JOINT INSTITUTE FOR NUCLEAR RESEARCH

[Dzhelepov](https://web.archive.org/web/20031204011713/http:/dzhelepov.jinr.ru/) laboratory of Nuclear Problems

**FINAL REPORT ON THE**

**START PROGRAMME**

SPD EventIndex

**Supervisor:**

Fedor Valerievich Prokoshin

**Student:**

Richard Gurtsiev, Russia  
North Ossetian State University

**Participation period:**

July 23 – September 15,

Summer Session 2024

Dubna, 2024

TABLE OF CONTENTS

Abstract 3

Introduction 4

History of development 4

SPD information ecosystem 5

Containerization 7

Automaic Deploy (CI-CD)\_\_\_\_\_\_\_\_ 8

Docker Swarm 10

Conclusion 12

Bibliography 13

Acknowledgments 14

АBSTRACT

The EventIndex system is being developed as a global catalog of events and event-level metadata for the SPD experiment at the NICA superconducting collider. EventIndex should provide indexing of event data, transfer of this information and writing to the database, access to it by users through interactive and asynchronous interfaces, as well as access to it by data processing and analysis programs through the API. Project development begins with the backend. Interfaces have been created: a command-line client and a graphical web interface for searching for information about events. A messaging service is being developed for asynchronous processing of requests to large amounts of data.

Currently, work is underway on the autonomous deployment of the project infrastructure. Technologies such as Docker Swarm and GitLab CI-CD for load balancing and orchestration of EventIndex project components are considered.

INTRODUCTION

The SPD experiment is a research project aimed for studying the physics of elementary particles and searching for new fundamental knowledge about the interactions and structure of particles. The main goal of this project is to analyze the data obtained from the experimental SPD detector installed at the NICA particle accelerator, created at the Joint Institute for Nuclear Research, located in the city of Dubna, Moscow region.

The SPD installation is a large-scale system of detectors capable of registering and analyzing various events that occur during collisions of nuclei producing elementary particles with high energy.

To store and process a large amount of data obtained during the SPD experiment, it will be necessary to create complex information systems. One of such systems should be an EventIndex - a catalog of physical events received from the installation or modeled for data analysis and processing.

HISTORY OF DEVELOPMENT

When developing the IS of the SPD experiment, it is assumed to widely use the experience gained in the development of similar systems for existing experiments. In particular, in the ATLAS experiment at the LHC accelerator, there is a similar catalog of events - the ATLAS EventIndex system. Each year, the experiment produces about 30 billion events, and about 100 billion are generated additionally. Event records are stored in files located on nodes of the GRID distributed computing system. The presence of such a huge amount of information determined the need to create a global catalog that would allow determining the location of each record for processing and analysis.

In the SPD experiment, data volumes of the same order as on ATLAS are expected with even more events. Despite the differences in the tasks set in these experiments, their data processing systems have similar features, which makes it possible to use similar solutions. SPD is supposed to use a similar data model and file organization, as well as a distributed data storage and processing system. Groups of statistically equivalent events are stored in files on disk or on magnetic tape. Each file usually contains from 1000 to 10000 events, depending on the format. Files are grouped into datasets, usually containing events related to a single data-taking session (Run). The main difference between the experiments is the absence of a trigger, instead, the initial selection of data will be carried out by an online filter based on machine learning methods.

SPD INFORMATION ECOSYSTEM

The architecture of the EventIndex project is based on a micro-service approach, which means that each component of the system is a separate service operating independently of the others. This approach provides flexibility, as each microservice can be updated and deployed independently of the others, which simplifies project management and improves system scalability. The system consists of three main blocks: the client part, the management system and data indexing.

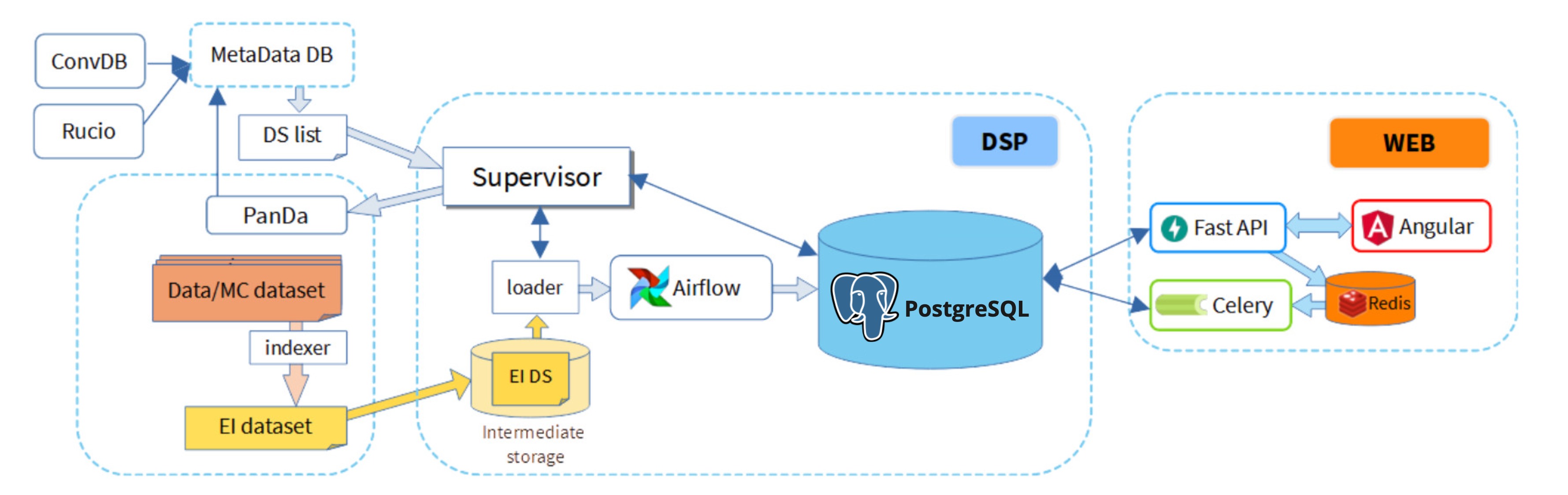


Figure 1: SPD EventIndex data flow architecture and schema

The first section, the client part of the system, is responsible for user interaction with the platform. It is implemented using several key technologies:

* **Angular** is included in the quality of the web interface development platform. Angular provides a dynamic and intuitive user interface that makes it easy to interact with the system through a web browser. This platform supports the implementation of complex user interfaces, providing high performance and ease of use.
* **FastAPI** used to create RESTful APIs that serve as an intermediate link between the client interface and the server part of the system. The Facet API provides fast and reliable data exchange between the frontend and backend, and is also responsible for processing requests from the user via Angular.

usage.

* **Celery and Redis** are these technologies implement asynchronous output of results. Celery manages tasks that require long-term processing, such as heavy calculations or working with large amounts of data. Red is, in turn, acts as a message broker and temporary storage for task results, providing high processing speed and minimal delays. As a result, users can receive the results of their queries without waiting for their processing to be completed.

These components together create a powerful and flexible client part of the SPD EventIndex, providing convenient and efficient access to data and system functionality.

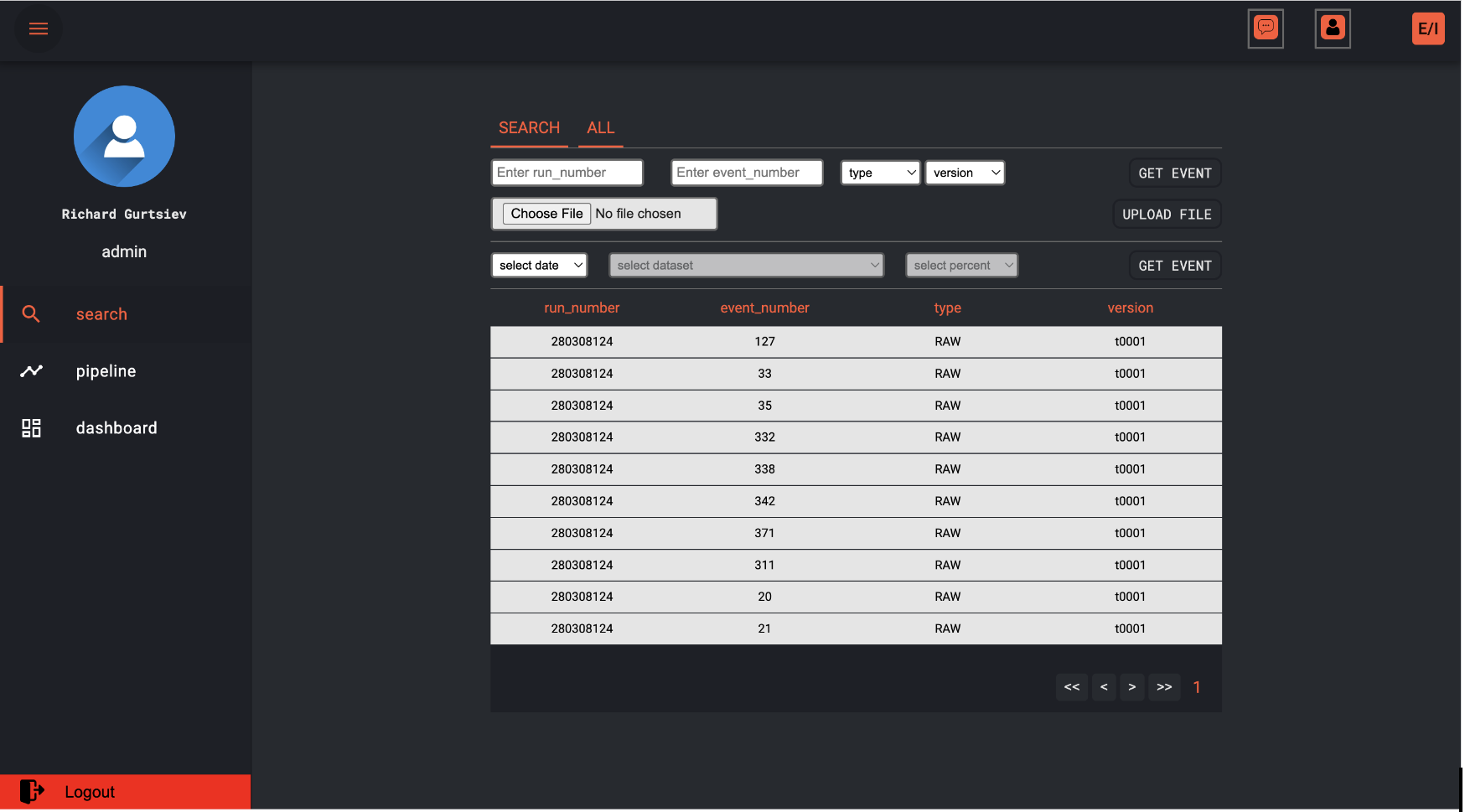


Figure 2: The main page of the web service

The second section is responsible for data management, ensuring reliable work with large amounts of information and performing complex analytical tasks. This section includes the following key technologies:

* **PostgreSQL** is the main database management system in the SPD EventIndex architecture. PostgreSQL provides reliable and scalable storage for all data processed by the system. It supports complex queries and provides high performance when working with large amounts of data.
* **AirFlow** a tool for organizing, planning and monitoring workflows. In SPD, EventIndex AirFlow is used to automate and coordinate ETL (Extract, Transform, Load) tasks — extracting data, converting it, and uploading it to storage. It provides the ability to create flexible and complex workflows that can be customized depending on the needs of the system.
* **Supervisor** a process management system that guarantees the smooth operation of all components of the data management and storage section. Supervisor automatically monitors and restarts processes if they fail, ensuring the stability and stability of the system.

These technologies together provide reliable and efficient data management in the SPD EventIndex, supporting high system performance, stability and scalability.

The third section is responsible for data indexing: it monitors the appearance of new datasets, the launch and control of indexing programs in a distributed data processing system, as well as uploading the results for import into EvenIndex. To perform these tasks, it is necessary to interact with other information systems of the experiment:

* **Rucio** a distributed data management system originally designed to work with large amounts of data on the LHC. At SPD EventIndex Ru, the cio is responsible for managing data replication, transferring data between different repositories, and complying with data retention policies.
* **Panda** a distributed computing management system that is used to work with a huge amount of data. Panda allows you to efficiently distribute computing tasks between different resources, ensuring that they are completed in optimal time.
* **Metadata IS** is being developed to collect, store and provide access to the physical metadata of the SPD experiment. These include information about datasets, including inheritance and links to task configurations that created these datasets. The use of metadata IS will make it possible to efficiently select data sets for indexing.

CONTAINERIZATION

Containerization is a technology that allows you to package an application and all its dependencies into a single, isolated container. Containers provide a lightweight alternative to virtual machines, allowing you to run multiple containers on the same host with minimal resource consumption.

Each component of the EventIndex project is isolated and can be developed, deployed and updated independently of the others. To ensure this independence and simplify the management of services, containerization is used. It allows you to package each service in a separate container with its dependencies and environment, which makes it independent of the main operating system and other elements.

Advantages of using containerization:

* **Isolation**: Each container is isolated from other containers and from the host, which prevents conflicts between applications.
* **Portability**: Containers can be easily transferred between different environments, whether in development, testing or production.
* **Scalability**: Containers are easily scaled, which simplifies load management in micro service applications.
* **Fast deployment**: With ready-made container samples, new application instances can be deployed in seconds.

At the moment, the basic images used by the EventIndex project are taken from the Docker Hub public repository, and after they are finalized and configured, they are stored in JINR GitLab Container Registry. This ensures stricter version control and security, since access to images is limited only to authorized users. This topic will be covered in more detail later, including CI/CD processes and image management in the EventIndex micro-service architecture.

AUTOMATIC DEPLOY (CI-CD)

CI-CD (Continuous Integration/Continuous Deployment) is the practice of automating the stages of software development, from writing code to deploying it.

**Continuous integration** (CI) involves frequently merging code added by developers into the main repository. Each such integration is accompanied by automatic testing and assembly, which allows you to quickly identify and eliminate errors.

**Continuous Deployment** (CD) is the process of automatically delivering modified code to various environments after successfully passing all CI checks. This approach speeds up the development process and reduces the risks associated with deploying new versions of the application. Service images after all successfully completed stages are saved in Container Registry, a private repository that provides centralized storage of containers.

This provides several important advantages:

* **Security:** Images are stored in a private repository, access to which is restricted only to authorized users. This allows you to protect confidential data and prevent unauthorized access to containers.
* **Version control:** GitLab Container Registry makes it easy to manage image versions, providing convenient access to any previous versions, as well as the ability to roll back to stable versions if necessary.
* **CI/CD integration:** Since containers are stored in GitLab, they can be used automatically in CI/CD processes. This simplifies and speeds up the deployment of new versions of system components.

The process of building Docker images within the CI/CD project EventIndex is automated using a YAML file. This file defines all the stages of the build and deployment, such as:

* Build: At this stage, a Docker image is created from the source code of the project.
* Test: The image is checked for errors using automated tests.
* Deploy: After successfully passing all the tests, the image is automatically deployed to a particular environment.

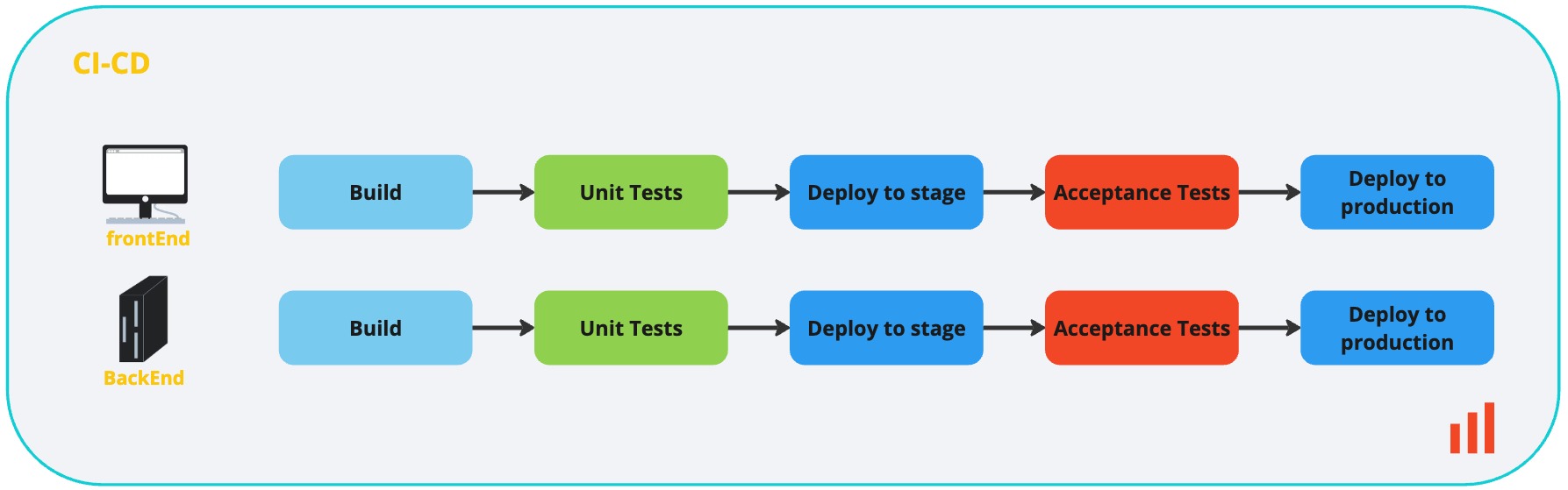


Figure 3: Pipeline for deploying the EventIndex project to the server

This is how the images of the EventIndex project components are collected autonomously in the Container Registry, going through all the testing stages before deployment on a virtual machine. This increases the level of security and simplifies container version management, which is especially important in the context of the microservice architecture of the project.

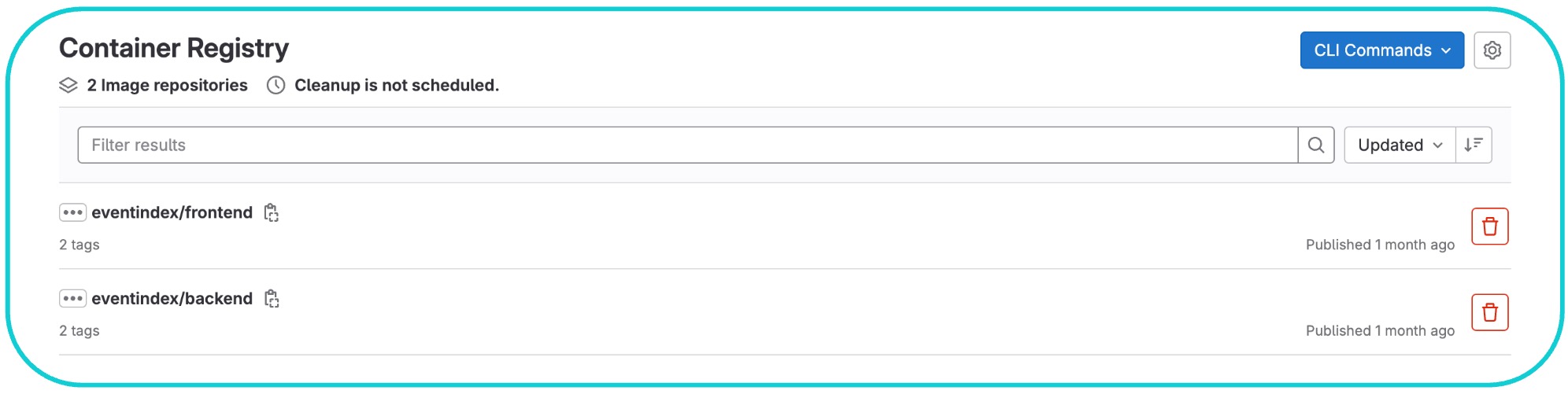


Figure 4: Images of the EventIndex project in the Container Registry

DOCKER SWARM

Docker Swarm is a clustering and orchestration solution provided by Docker. It allows you to create and manage multiple Docker nodes, which can be physical or virtual machines running Docker Engine. These nodes communicate with each other using the Docker API, which allows them to work together as a single large-scale Docker cluster.

The main entities in Docker Swarm:

* **Node** are the virtual machines on which Docker is installed. There are manager and worker nodes. The node manager manages worker nodes. She is responsible for creating/updating/deleting services on all work nodes, as well as for scaling and maintaining them in the required state. Worker nodes are used only to perform assigned tasks and cannot manage the cluster.
* **Stack** is a set of services that are logically interconnected. Stack (services) parts can be located on the same node or on different ones.
* **Service** is what stack consists of. Service is a description of which containers will be created.
* **Task** is a directly created container that Docker created based on a detailed description of the service. Swarm will monitor the status of the container and, if necessary, restart it or move it to another node.

Advantages of Docker Swarm:

* **High availability**: Docker Swarm automatically replicates services, distributing them across available cluster nodes, ensuring fault tolerance.
* **Horizontal scaling**: It is easy to scale services by increasing or decreasing the number of container instances (replicas).
* **Rolling updates**: The ability to smoothly update services without interrupting their work, minimizing the "downtime" of the project.

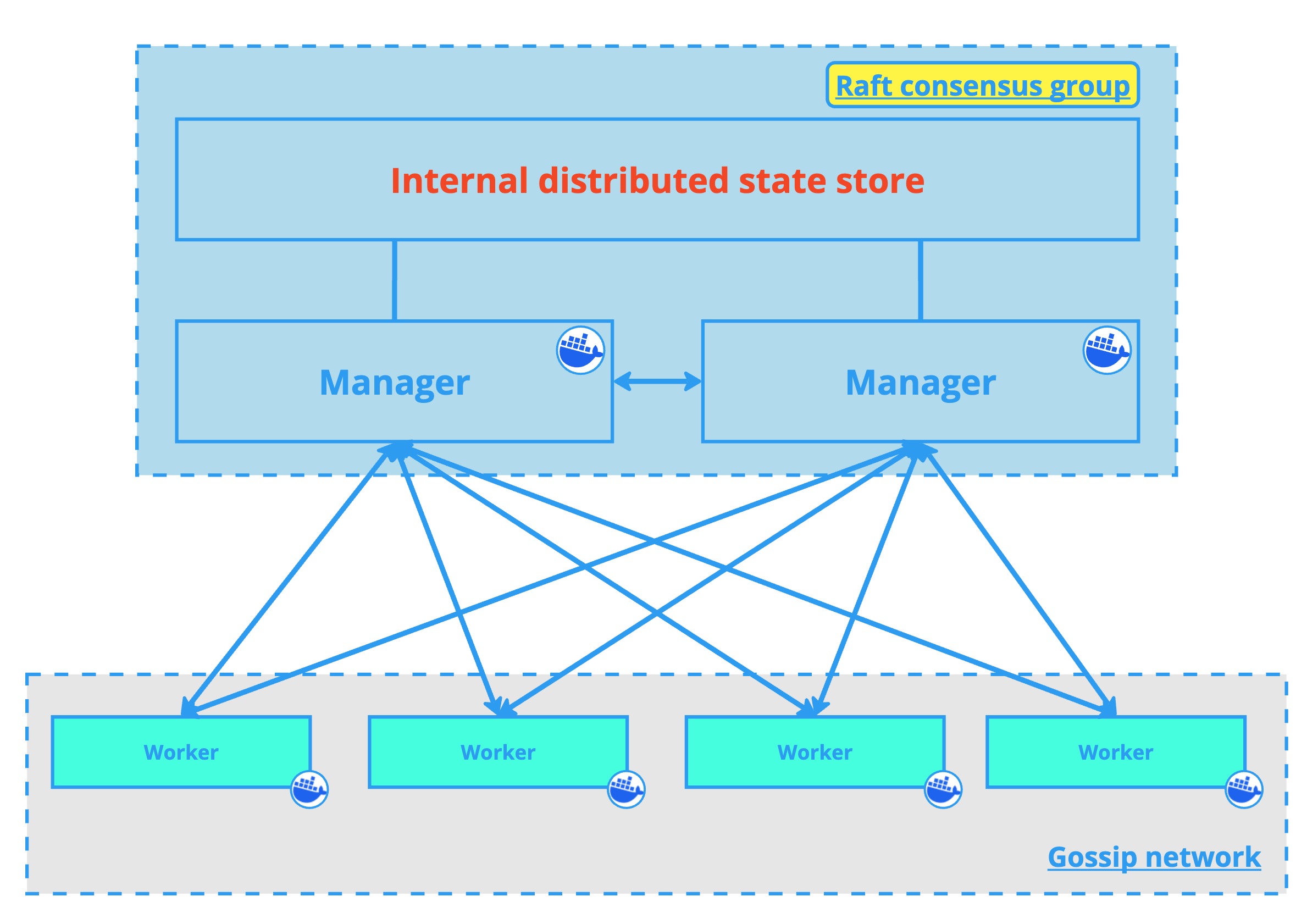


Figure 5: Relationship between worker and manager nodes

Using Docker Swarm as part of the EventIndex project provided the opportunity to orchestrate containers collected and stored in the Container Registry. Now, when the service traffic increases, the incoming load is automatically distributed among all available replicas of containers. This helps to prevent overloading of individual containers and ensures an even distribution of requests across the entire infrastructure. Also, in case of problems on one of the nodes, Swarm will move containers to other nodes, ensuring the continuity of the project.

Currently, the project is hosted on JINR servers, which users can access and interact with.

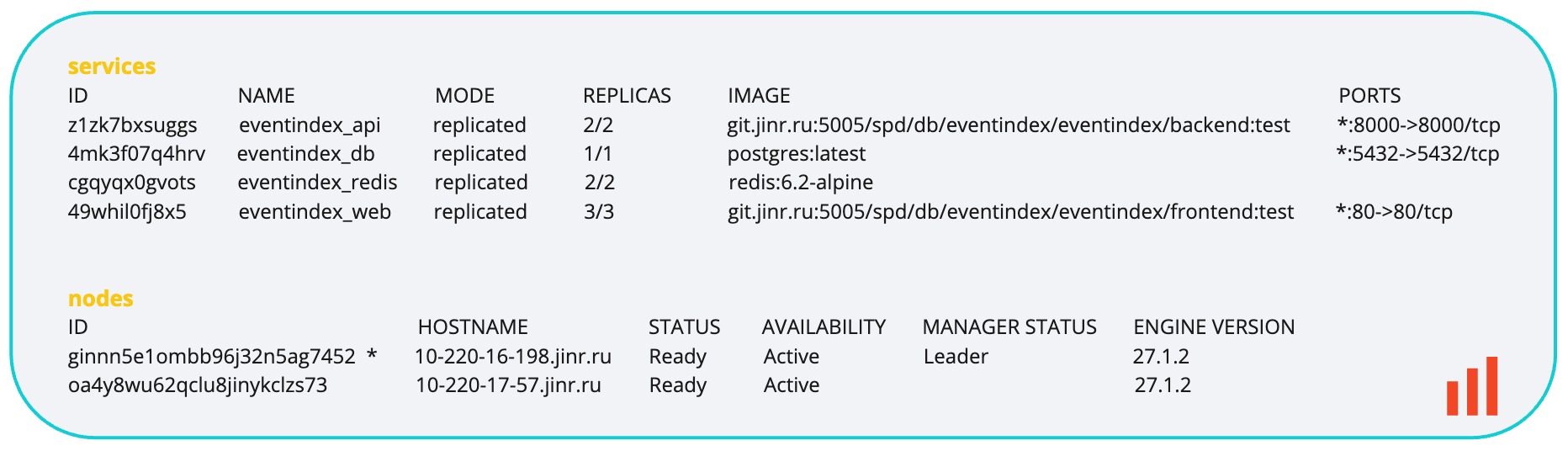


Figure 6: Cluster nodes and services

CONCLUSION

In the course of further development of the EventIndex project, the following tasks are expected to be completed:

1. Development of load tests for data processing.
2. Development of mechanisms for transmitting "EventIndex" data obtained by indexing files located on remote nodes of a computer network.
3. Development of a supervisor — software for managing data collection and import into EventIndex.
4. We will consider other container management options.

The implementation of these tasks will be carried out in cooperation with the developers of other information systems of the SPD experiment and other experiments at the NICA accelerator complex.

BIBLIOGRAPHY

* [The ATLAS EventIndex: A BigData Catalogue for All ATLAS Experiment Events](https://arxiv.org/ftp/arxiv/papers/2211/2211.08293.pdf), Computing and Software for Big Science (2023) 7:2
* [Information systems for data taking and data processing, SPD collaboration meeting, 2023](https://indico.jinr.ru/event/3575/contributions/20669/attachments/15249/25742/2023-04-26.Information%20systems%20for%20data%20taking%20and%20data%20processing.pdf)
* [TechnicalDesignReport\_SPD2023](http://spd.jinr.ru/wp-content/uploads/2023/03/TechnicalDesignReport_SPD2023.pdf)
* [Docker: Documentation](https://docs.docker.com/?_gl=1*7153c8*_gcl_au*MTY0NTE0NTY4Mi4xNzI2MjI0NDU1*_ga*NTk0NDAyNTM4LjE2OTkxNjA2OTY.*_ga_XJWPQMJYHQ*MTcyNjIyNDQ1NS41NS4xLjE3MjYyMjQ0NTYuNTkuMC4w)
* [Docker Swarm: Docker Swarm for the youngest / Hubr](https://habr.com/ru/articles/659813/)
* [CI-CD: Documentation](https://docs.gitlab.co.jp/ee/ci/#:~:text=GitLab CI%2FCD is a tool,simplify and scale software development)

ACKNOWLEDGMENTS

The author expresses gratitude to Fedor Prokoshin for providing comprehensive guidance and moral support in the process of work, as well as for technical support and valuable recommendations. I am also grateful to the organizers of START for the opportunity to complete the program.