Microservices for Systematic Profiling and Monitoring of the Refactoring

UNIVERSITY OF BIRMINGHAM

Alexander Mazurov, Ben Couturier

LHCb Performance and **R**egression Tests (**LHCbPR**) - systematize profiling that helps developers to evaluate how their recent **code changes** behave in provided test cases for **different setup environments**.

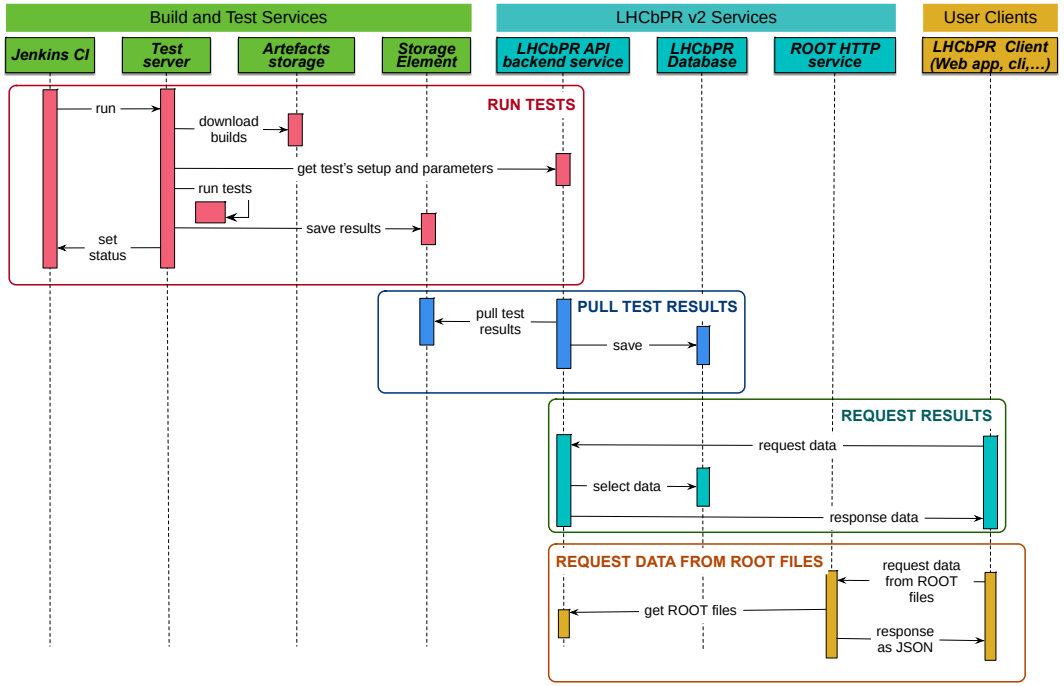
Main use cases

- Physics performance
  - Histogram comparison
  - Trend analysis for selected attribute.
- Monitor regression in memory and CPU consumption

Possible setup environments

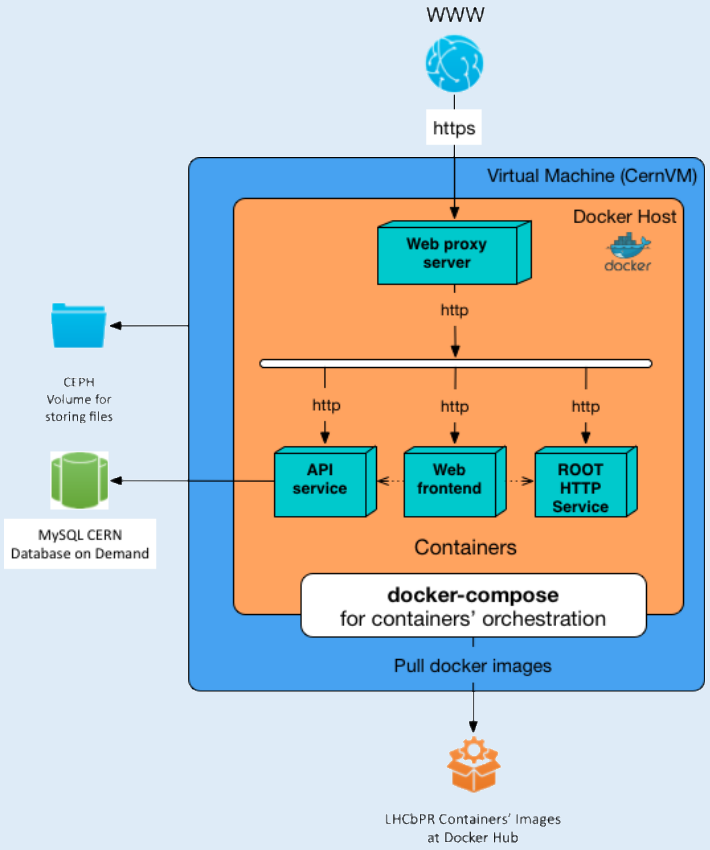
- Versions of application
- Compiler versions
- Operating Systems (SLC6, CentOS7)
- Architecture (x86_64, x86)
- Build system (CMT or CMake)

- **LHCbPR** not coupled to the LHCb software stack and can be **adapted for other experiments and projects**
- We are working on extending repository of **web components** and analysis modules for web frontend.
- Easy to develop **new clients** for API service.

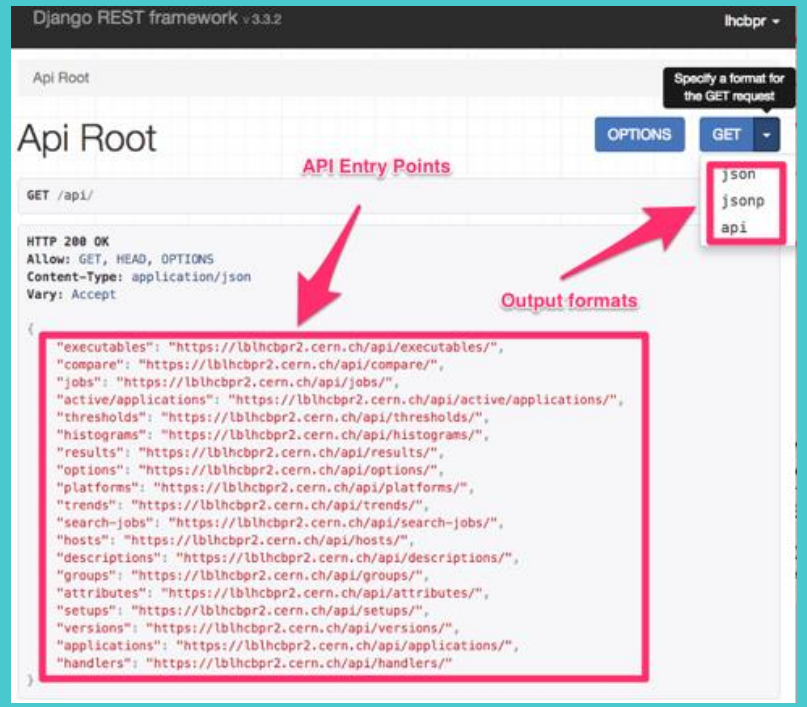


1. **Build and Test Services**
   - **Continuous Integration (CI) Service** – schedule and initiate test runs
   - **Artifacts Storage**– store projects builds for different configurations
   - **Test service** – read LHCbPR configuration for tests, download the corresponding builds, execute tests and transfer it to the Storage Element
   - **Storage Element** – virtual storage for jobs output with the interface to quite diverse real storage systems like grid storage.
2. **LHCbPR v2**
   - **Database –** relational database for job descriptions and job outputs. We use **MySQL**, but it can be any other.
   - **REST API service** – provides REST access to the database and adds some business logic for special API requests. Technologies: **python**, **Django** + **REST Framework**.
   - **ROOT HTTP service –** helper service for returning content of ROOT files in JSON format. Relies on ROOT TBufferJSON.ConvertToJSON functionality. Technologies: **Flask** , **ROOT**.
3. **User Clients**
   - Users can create any data handling client that use LHCbPR REST API: web applications, scripts
   - We created web frontend for visualizing regression tests' results. Technologies: **javascript**, **angular framework; nodejs and gulp** for development.

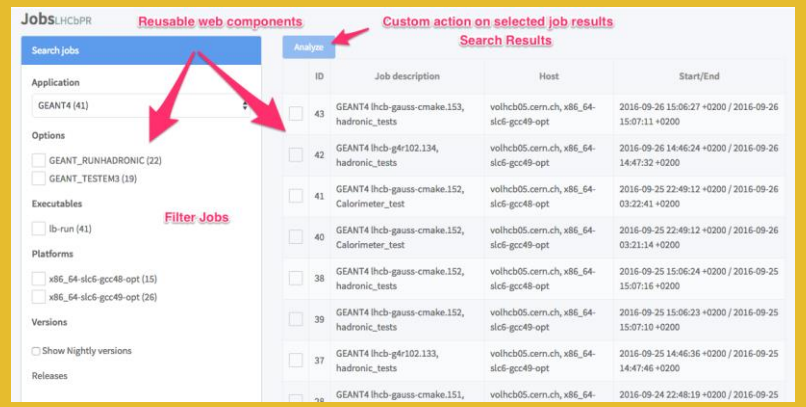
Deployment

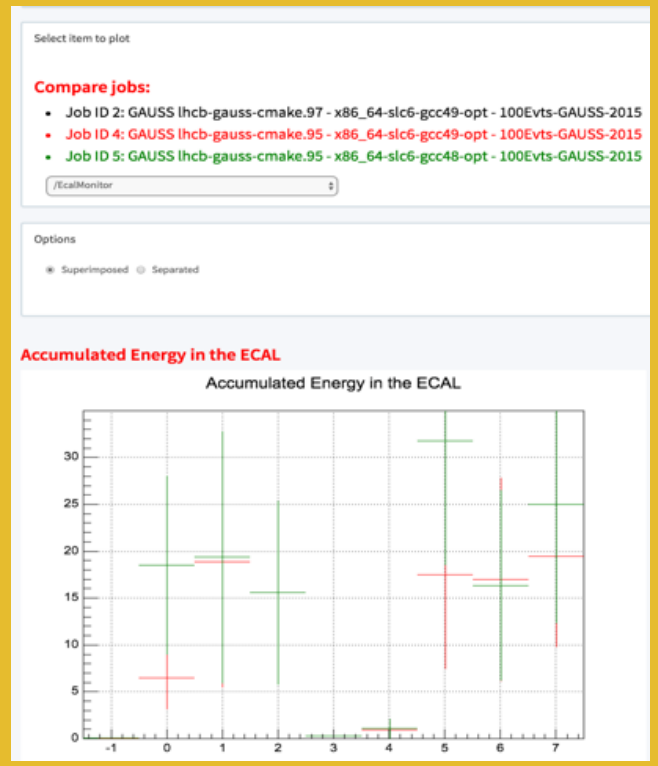


API Service



Web Frontend





- Web frontend is a javascript single-page application that is composed of **analysis modules** for presenting specific logic and views for inspecting test results.
- Each analysis module is an **application extension** and can be simply added or removed without breaking the main application
- Common **web components** are provided for building modules. For example, search jobs and draw histograms.