ORT Server

Implementation Update

Martin Nonnenmacher
06.03.2024

01

Motivation

Motivation

Building a scalable ORT service

Scalability

Decouple from limitations of CI Pipelines

ORT is currently mainly deployed using CI Pipelines like Jenkins, GitHub Actions, or Azure Pipelines. This limits scaling options as these pipelines are not designed to run a service.

01

REST API

REST API eases integration with other services

The REST API (with OpenAPI Spec) allows to integrate the ORT Server with other services. For example, the CI integrations can simply call the API instead of implementing a complex ORT workflow.

02

Database

Central database for all ORT results

Instead of separate ORT result files for each run, the server stores all results in a central database. This enables performance optimizations and data analysis, for example, to find out which projects use a specific dependency.

03

Credentials

Credential management via API

Running a central ORT service also requires to manage the required credentials for analyzing the projects. With ORT Server, users can manage their credentials via the REST API.

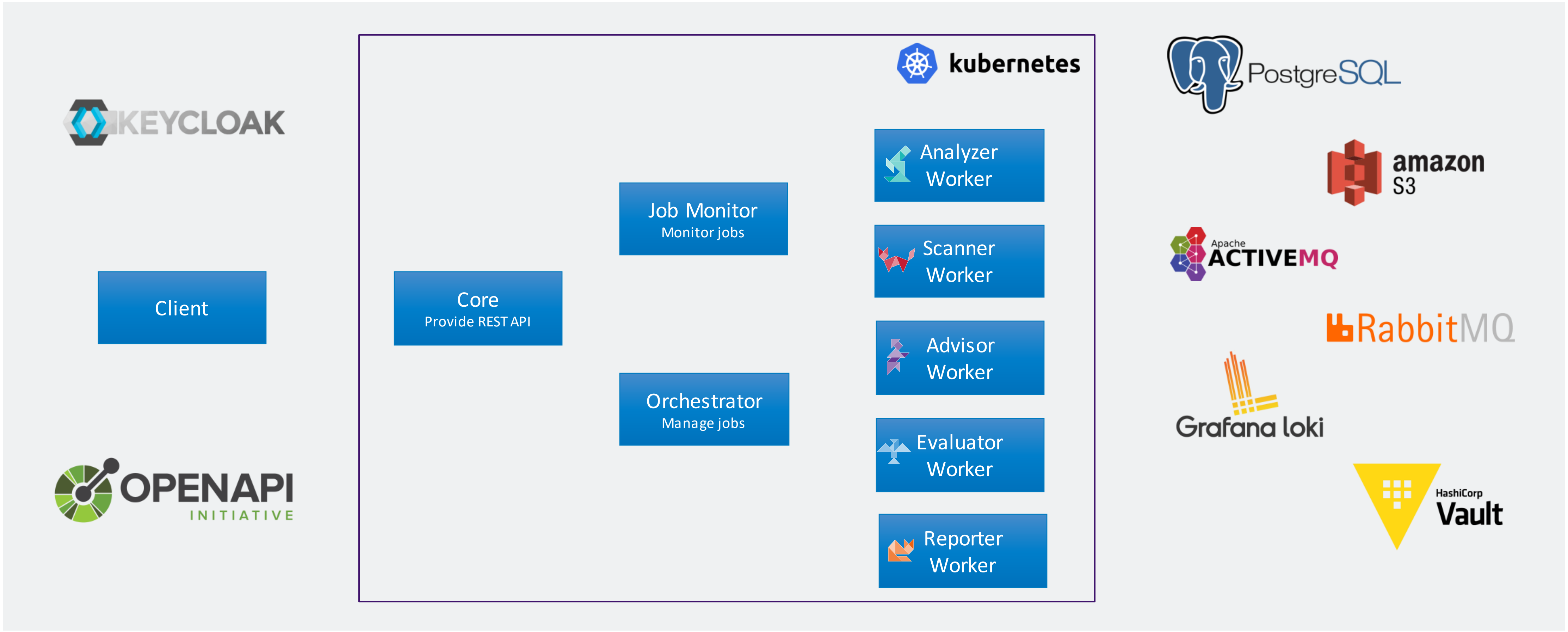
04

BOSCH

02

Architecture

Architecture
Components



BOSCH

Architecture
Adaptable

Adaptable Components

Components that can be replaced by similar tools

- Binary data: S3, PostgreSQL
- Messaging: ActiveMQ, RabbitMQ
- Secrets: HashiCorp Vault
- Log Aggregator: Loki

Required Components

Components that are mandatory

- Keycloak: authentication and role management
- PostgreSQL: relational data
- Docker: application containers

BOSCH

Architecture
Content Hierarchy

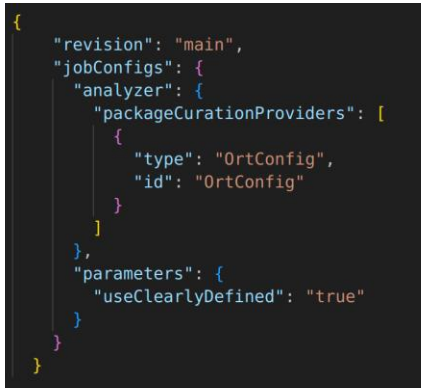
- Content hierarchy
  - Organizations: Group of products
  - Products: Group of repositories
  - Repository: Single source code repository
- User roles on each hierarchy level
  - Admin: All permissions
  - Writer: Edit content
  - Reader: Read content
- Credential management
  - Secrets can be defined on each level of the hierarchy
  - Secrets are stored in a dedicated service (not the database)

Architecture
Access to protected resources

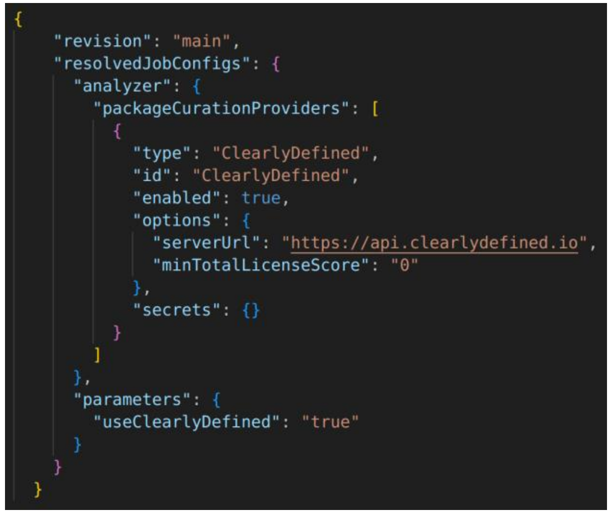
- Infrastructure services
  - Define external services by URL
  - Define credentials required to access the service
- Environment services
  - Define package manager specific configuration, like required Maven repositories
  - Associate package managers with infrastructure services
  - Generate package manager specific configuration files, like Maven settings.xml
  - Define required environment variables
- .ort.env.yml
  - Both service types can be configured in an .ort.env.yml file in the repository
  - Contains only references to secrets

Architecture
Company specific API

- Maybe not all features of the API should be available to users when triggering a run.
  - For example, only a predefined list of curation providers could be allowed.
- Company specific rules can be implemented in a Kotlin script that modifies API requests.
- Options can be simplified using a parameters map in the request.



```
{
  "revision": "main",
  "jobConfigs": {
    "analyzer": {
      "packageCurationProviders": [
        {
          "type": "OrtConfig",
          "id": "OrtConfig"
        }
      ]
    },
    "parameters": {
      "useClearlyDefined": "true"
    }
  }
}
```



```
{
  "revision": "main",
  "resolvedJobConfigs": {
    "analyzer": {
      "packageCurationProviders": [
        {
          "type": "ClearlyDefined",
          "id": "ClearlyDefined",
          "enabled": true,
          "options": {
            "serverUrl": "https://api.clearlydefined.io",
            "minTotalLicenseScore": "0"
          },
          "secrets": {}
        }
      ]
    },
    "parameters": {
      "useClearlyDefined": "true"
    }
  }
}
```

03

Outlook

Outlook
Plans for the next year

Getting Started

Make it easy to get started with ORT Server

Improve the documentation for deploying and customizing ORT Server and provide deployment templates like a Helm chart.

01

Frontend

Web frontend to manage repositories and view results

It is planned to develop a web frontend in collaboration with DoubleOpen. The frontend could be used to manage content like repositories or secrets, to view results, or to simplify the curation of results.

02

Integrations

Tooling to develop CI integrations

Provide a containerized client that can be used by CI integrations like GitHub actions to simplify authentication and API calls.

03

Performance

Use the potential for performance optimizations

The first version of the ORT Server does not fully utilize the potential for performance optimizations as the focus was on providing a stable solution. There is much potential for future versions to improve the performance.

04

04

Questions?

BOSCH