# ARAS Documentation

***Release 0.1.0***

**Ilaria Pigazzini, Cezar Sas, Alessandro Tundo, Matteo Vaghi**

**Jul 22, 2018**

# Contents:


**1 Synopsis** **2**

**2 Architecture overview** **3**
- 2.1 Analyses configurator service . . . 4
- 2.2 Analyses executor service . . . 4
- 2.3 Notifications service . . . 4
- 2.4 Projects service . . . 4
- 2.5 Reports service . . . 4
- 2.6 Config server . . . 4
- 2.7 Eureka server . . . 4
- 2.8 Gateway . . . 5

**3 Development** **6**
- 3.1 How to run . . . 6

**4 Authors** **7**

**5 License** **8**

You can visualize the build status here and the code quality here.

# CHAPTER 1

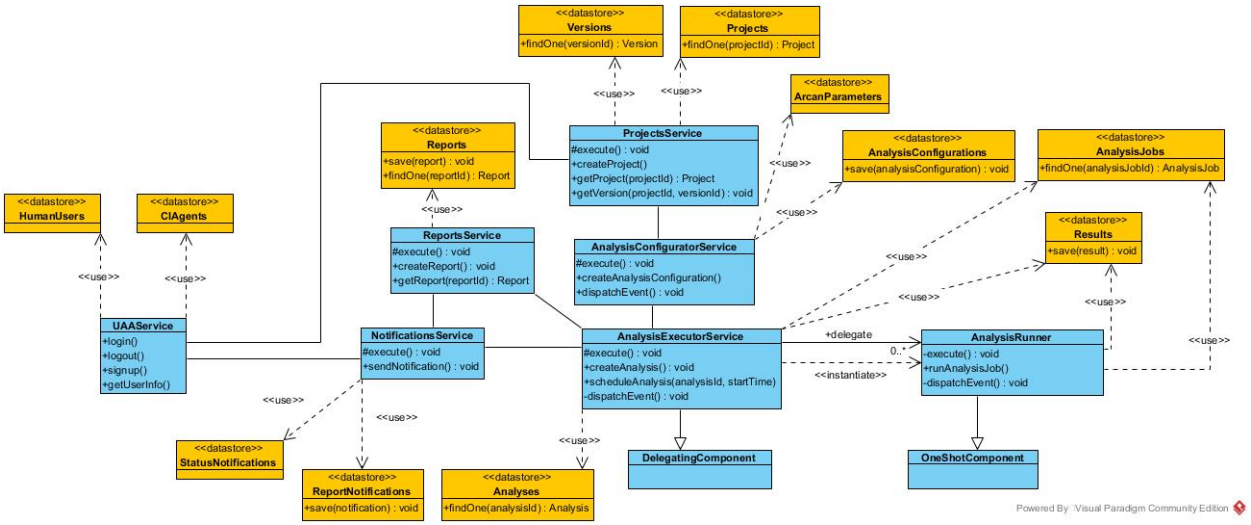
## Synopsis

A lot of software analysis tools are available to monitor the various aspects of a system. One interesting facet is the design quality. A good design leads to evolvability, maintainability of the system, its availability, its security and cost reduction. When there is a lack of design, or the system is made up by poor design choices, the architecture of the system could be subjected to architectural anomalies: ARAS offers a service to run static analyses on software and inspect the retrieved results, to support developers and mainteiners during the quest for those anomalies. In particular, ARAS aim is to expose the existing software Arcan as a web service. Arcan, which is a static analysis tool, analyses compiled Java projects in order to detect architectural flaws, in particular the ones called Architectural Smells (AS). At the moment Arcan offers a desktop user interface and runs locally. ARAS will not extend Arcan domain features, but it will use it as a black-box.

CHAPTER 2

# Architecture overview

The following section gives a brief overview on the architecture.




The components which are already available are:

1. analyses-configurator-service
2. analyses-executor-service
3. notifications-service
4. projects-service
5. reports-service
6. config-server
7. eureka-server
8. gateway

Moreover, the infrastructure provides Apache Kafka (with Zookeeper) as message broker shared among services.

## 2.1 Analyses configurator service

The analyses-configurator-service is a SpringBoot-based microservice that relies on a MongoDB storage engine. It encapsulates the logic to manage analysis configurations. It also exposes a RESTful API and publishes/consumes notable events to/from Kafka.

## 2.2 Analyses executor service

The analyses-executor-service is a SpringBoot-based microservice that relies on a MongoDB storage engine. It encapsulates the logic to manage analysis scheduling and execution. Moreover, it also exposes a RESTful API and publishes/consumes notable events to/from Kafka.

## 2.3 Notifications service

The analyses-executor-service is a SpringBoot-based microservice that relies on a MongoDB storage engine. It encapsulates the logic to manage and send notifications. It also exposes a RESTful API and publishes and consumes notable events to/from Kafka.

## 2.4 Projects service

The analyses-configurator-service is a SpringBoot-based microservice that relies on a MongoDB storage engine. It encapsulates the logic to manage projects and their versions. It also exposes a RESTful API and publishes/consumes notable events to/from Kafka.

## 2.5 Reports service

The analyses-configurator-service is a SpringBoot-based microservice that relies on a MongoDB storage engine. It encapsulates the logic to manage reports. It also exposes a RESTful API and publishes/consumes notable events to/from Kafka.

## 2.6 Config server

The config-server is a SpringBoot-based application which acts as a configuration server. It holds configuration files for all the services.

## 2.7 Eureka server

The eureka-sever is a SpringBoot-based application which acts as a service registry enabling service discovery for the whole architecture.

## 2.8 Gateway

The gateway is a SpringBoot-based application which acts as an API Gateway. It is the entrypoint for the clients. It performs to downstream services. Hereafter, the gateway will exploit a User Account and Authentication (UAA) Server to perform authentication and authorization checks.

CHAPTER 3

# Development

The development environment is provided through Docker containers. The `docker-compose.yml` file contains the common services definition between production and development. The `docker-compose.override.yml` file holds the additional definitions for the development.

Subsequently, a `docker-compose.prod.yml` file will be added in order to provide production configurations for a Docker swarm mode deploy.

## 3.1 How to run

Minimum requirements:

- Docker Engine 17.09.0+
- Docker Compose 1.8+

Since there are shared dependencies among services, it is strongly suggested (read it as *you have to*) start them in a specific sequence.

First, copy `.env.example` to `.env` and substitute the values accordingly to your needs.

Execute the following steps:

```
$ docker-compose build
$ docker-compose up -d config-server kafka
$ docker-compose up -d eureka-server
$ docker-compose up -d notifications-service analyses-configurator-service analyses-
↪executor-service projects-service reports-service gateway
```

CHAPTER 4

# Authors

- **Alessandro Tundo** - aletundo
- **Ilaria Pigazzini** - ipiga94
- **Matteo Vaghi** - oet93
- **Cezar Sas** - SasCezar

CHAPTER 5

# License

This project is licensed under the AGPLv3+. See the here for further details.