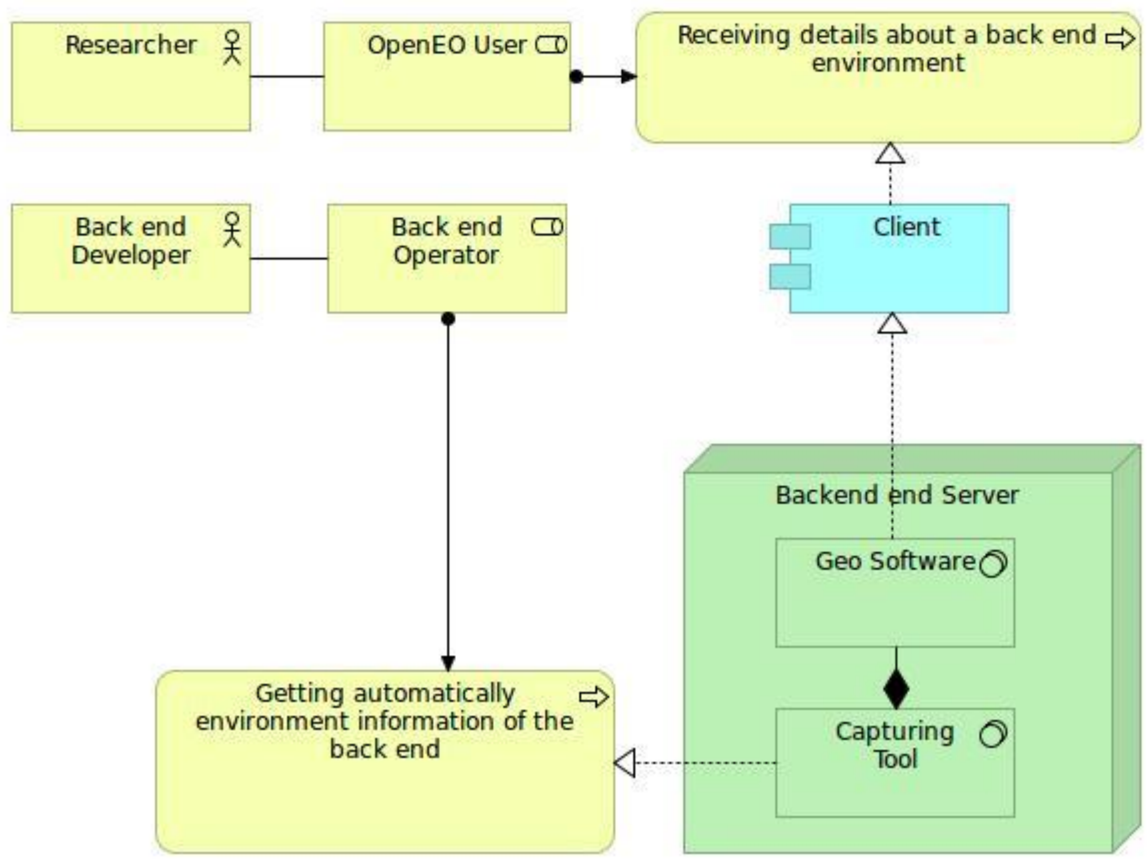
The openEO project creates a communication standard for data providers and users to enable them a way to use every back end they want to, without having to change their code. Therefore, it is a great opportunity to improve reproducibility in the earth observation science. The goal of this master thesis is to provide possible first steps towards making earth observation science reproducible by using the technology provided by the openEO project. Three Use Cases got defined and are described in the following sections.

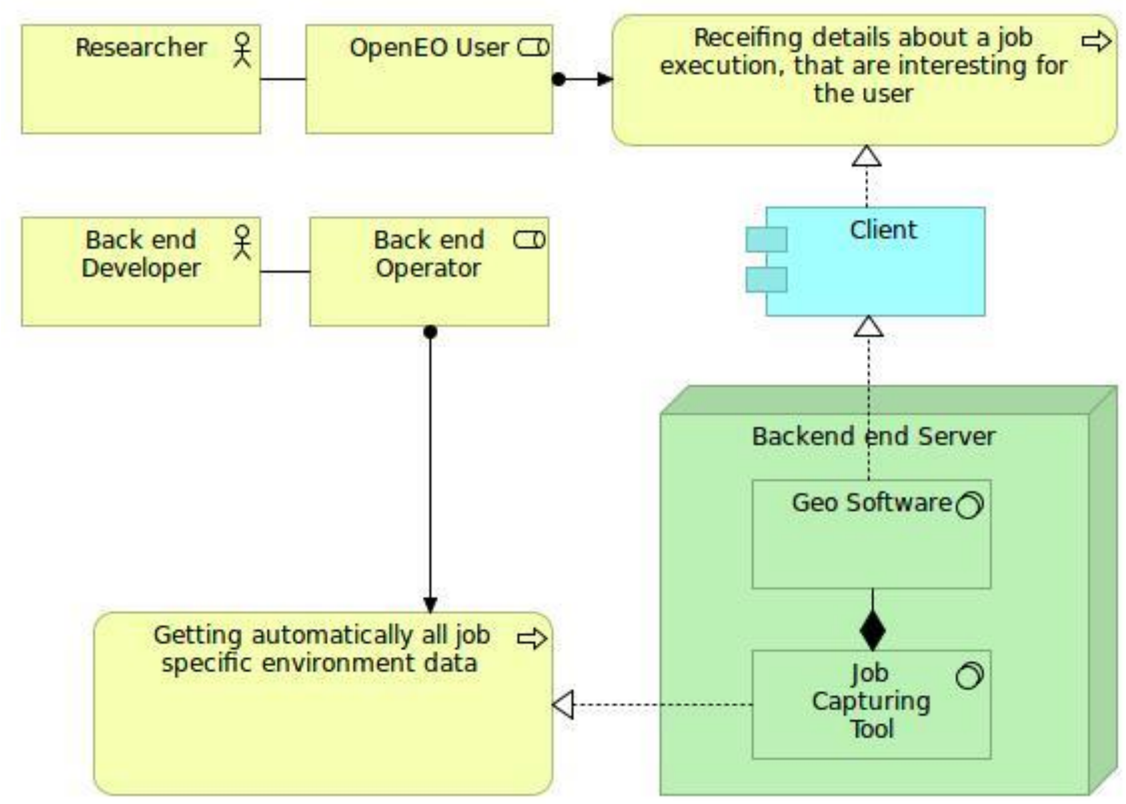
Capturing the Back-end Environment


Researcher
OpenEO User
Receiving details about a back end environment
Back end Developer
Back end Operator
Client
Backend end Server
Geo Software
Capturing Tool
Getting automatically environment information of the back end

In order to make a job execution in the openEO project reproducible, the environment of the back end has to be captured and persisted. It has to be specified in detail and should only contain job independent information about the backend like the used operating system, programming language or used openEO API version. The process of capturing has to be applied automatically on each change of the back end environments. To keep track of the changes of the server and the current status, a back end version shall be introduced, which has to be updated for every change made on the back end. The goal of this master thesis is to create a possibility to implement this behavior on one backend or to implement even an independent tool that can be used for every backend to capture the environment and create a version number in a standardized way.
The above section was describing the use case from the providers perspective and the other part is from the users perspective and should give the user the possibility to receive the back end details. Here only data have to be shown that are not a security threat to the back end provider, but are relevant to the user. Therefore a new GET endpoint has to be introduced into the openEO API like “/version” or it can be added to the “/capabilities” endpoint. This can help the users to compare different backends or help to decide, which they wants to use. It has to be defined, what job independent back end information is of interest for a typical EO user.

Methods
I am implementing a back end independent tool to capture the job independent environment data. So, every back end provider can use it to automatically generate a back end version in a standardized way. Therefore, a definition of what must be saved has to be introduced. This tool and endpoint will be evaluated on a python back end, based on the EODC OpenShift back end.
At the moment I am doing a scientific literature research and I am questioning colleagues, to get an overview what EO scientists currently write about their tools, to get a better understanding what is important to know about a back end.

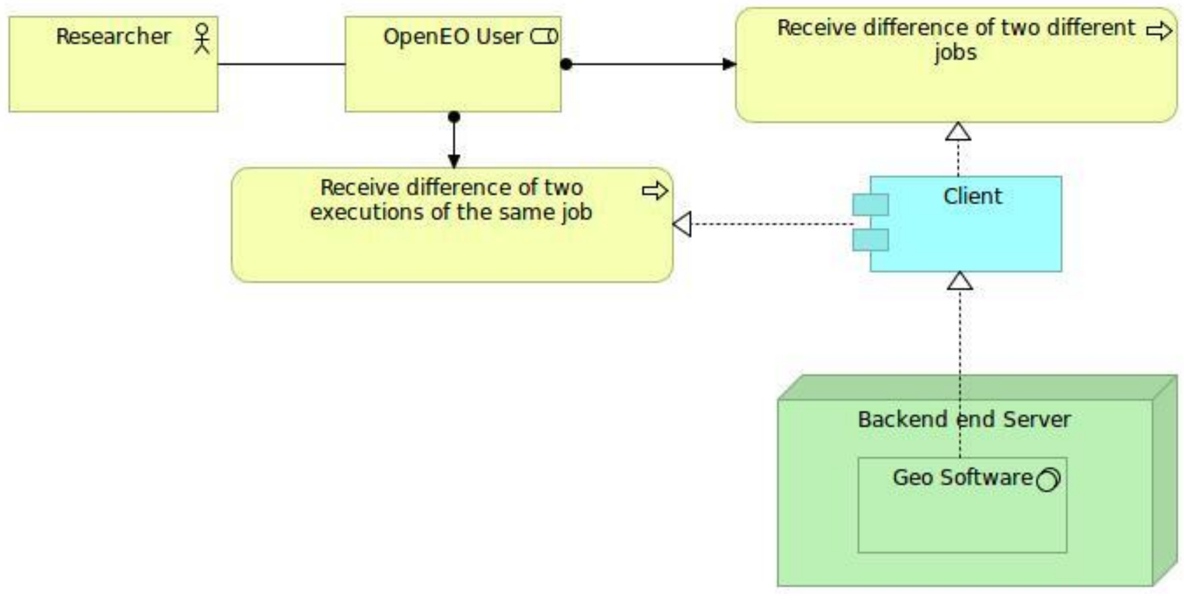
Capturing job dependent Environments


Researcher
OpenEO User
Receifing details about a job execution, that are interesting for the user
Back end Developer
Back end Operator
Client
Backend end Server
Geo Software
Job Capturing Tool
Getting automatically all job specific environment data

This use case is like the first one, but is only about job dependent environment information. The back end provider shall automatically get provenance data about the job execution, like used software packages and their versions. For the user additional information on a job execution must be introduced. Calling the job endpoint, shall result in additional information interesting to the typical EO user like calculation methods and e.g. the used GDAL version. So other than the back end provider, who gets automatically all provenance data, the users only get filtered and for them relevant provenance data. The motivation for this is to add transparency of the job processing for the users, so that researchers can describe their processes in more detail.

Methods
I am implementing the job specific capturing for the python back end based on the EODC OpenShift back end using noworkflow, which can be used for every python based back end. I also provide an overview of other technologies that can be used by other back ends.
Like in the first use case I am reading through papers and ask colleagues about what details are interesting for them about a job execution.

Getting differences of job executions


Researcher
OpenEO User
Receive difference of two different jobs
Receive difference of two executions of the same job
Client
Backend end Server
Geo Software

The third use case is dedicated to the openEO users. The goal is to add a possibility for users of openEO to compare different jobs not only by their results, but also on the way they were executed. There are two job comparisons considered within this master thesis. The first one is a comparison between the same job that gets executed several times. That is why an endpoint shall be introduced that makes it possible for users to compare different runs of the same job (same job id). The other comparison is between a job execution and different job executions (different job ids). To achieve this also in a standardized way, there has to be a set of measurements defined, which are considered for the comparison. In the current approach each measurement gets validated separately so that the user can distinguish the parts of the compared job execution that differ from the original execution.

Methods

I am again implementing this on my own python back end based on the EODC OpenShift backend. The captured data from the other two use cases are used for the validation. Two new endpoints shall be added to the openEO API for the comparison of different executions of one job and for the comparison of two or more different jobs. The captured data and validation technique must be defined, so that other back ends can make the validations in a similar way.

Example (compare different jobs)

Request: POST /jobs/<id>/diff
Body: {'job_ids': ['id1', 'id2', 'id3']}
Response:

```
{
  base_job_id: '<id>'
  results: [{
  "job_id": "id1",
  "validation": {
            "input_data": "EQUAL",
            "output_data": "DIFFER",
            "backend_version": "EQUAL",
            "code": "DIFFER",
            "code_env": "EQUAL",
            "job_id": "EQUAL",
            "openeo_api": "EQUAL",
            "process_graph": "DIFFER"
            }
    }
… ]
}
```