TELESPAZIO
a LEONARDO and THALES company

Earth Observation Exploitation Platform Common Architecture (EOEPCA+)

EOEPCA+ Release 2.0-rc1

James Hinton

17th July 2025

EOEPCA
BETTER ACCESS TO EARTH OBSERVATION

esa

Context – Exploitation Platform

Transforming Data to Actionable Information

Virtual analysis environment

Data

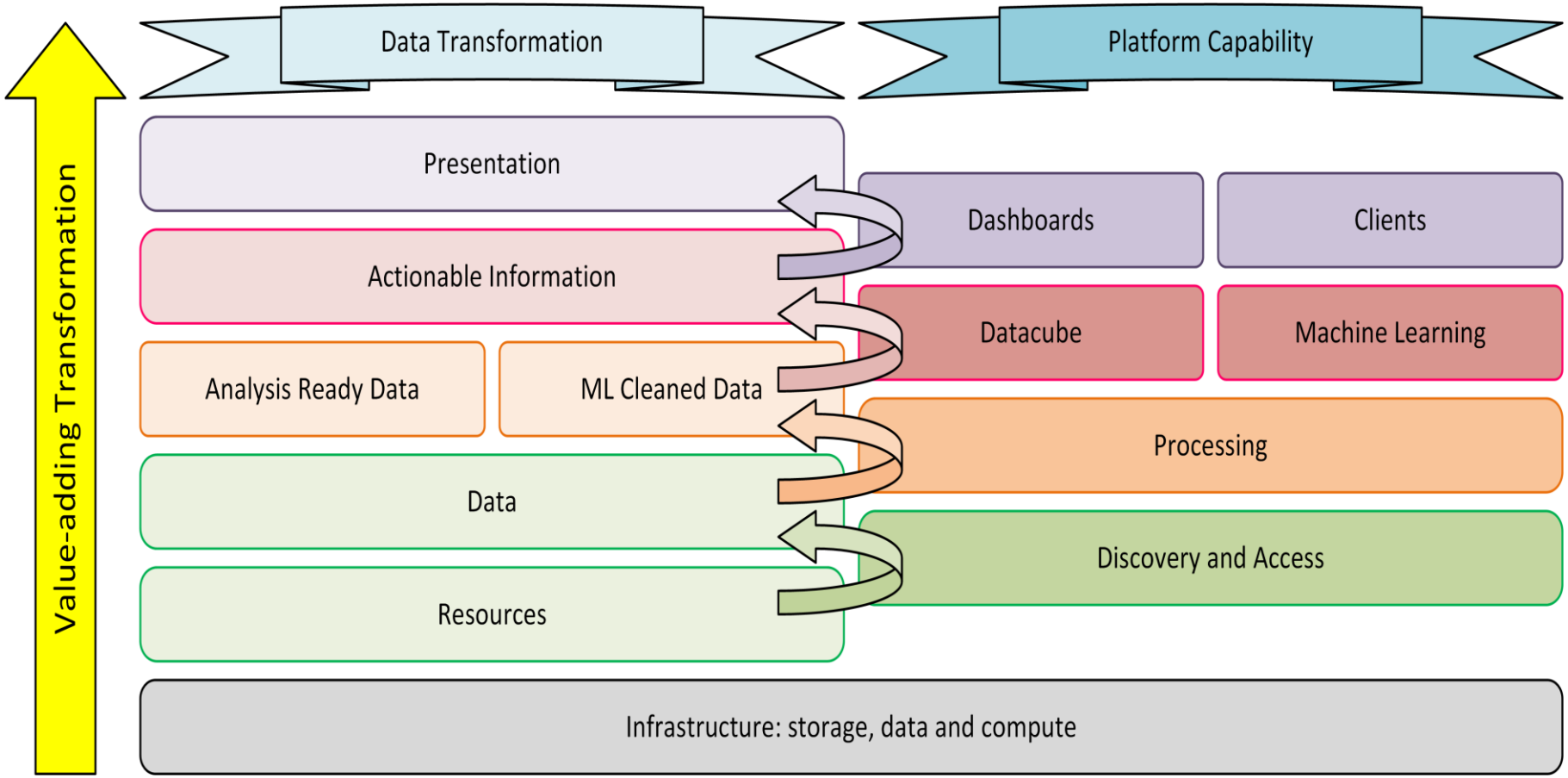
Compute

Tooling

Collaboration

Sharing

Publishing



Aims and Objectives

Problem

Many platforms in a fragmented ecosystem

Difficult for users to exploit their complementary offerings

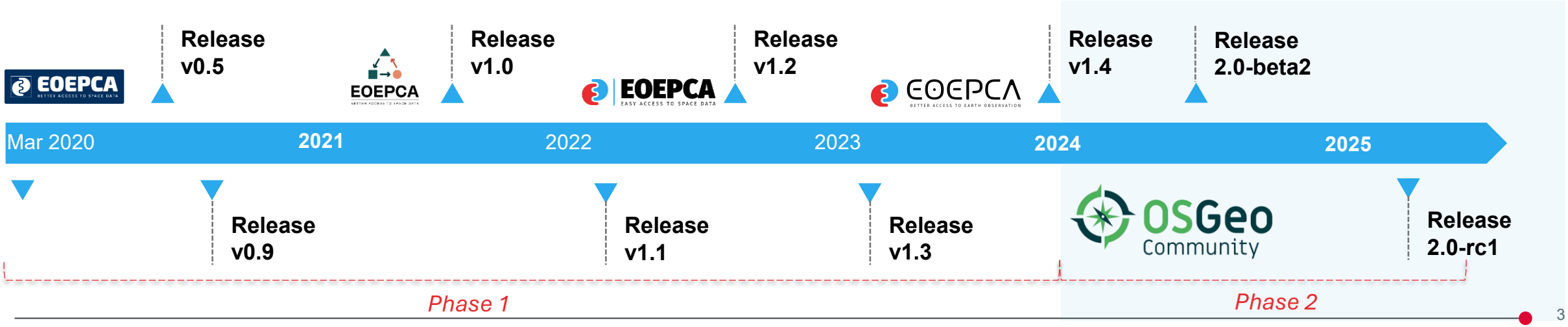
Our Approach

Common Architecture

- **Open Standards**
- Enabling **Federation** among EO cloud platform offerings
- Promote and develop **Interoperability** standards

Reference Implementation

- **Open Source**
- Avoid further fragmentation
- **Reusable Building Blocks**
- Reduce development costs



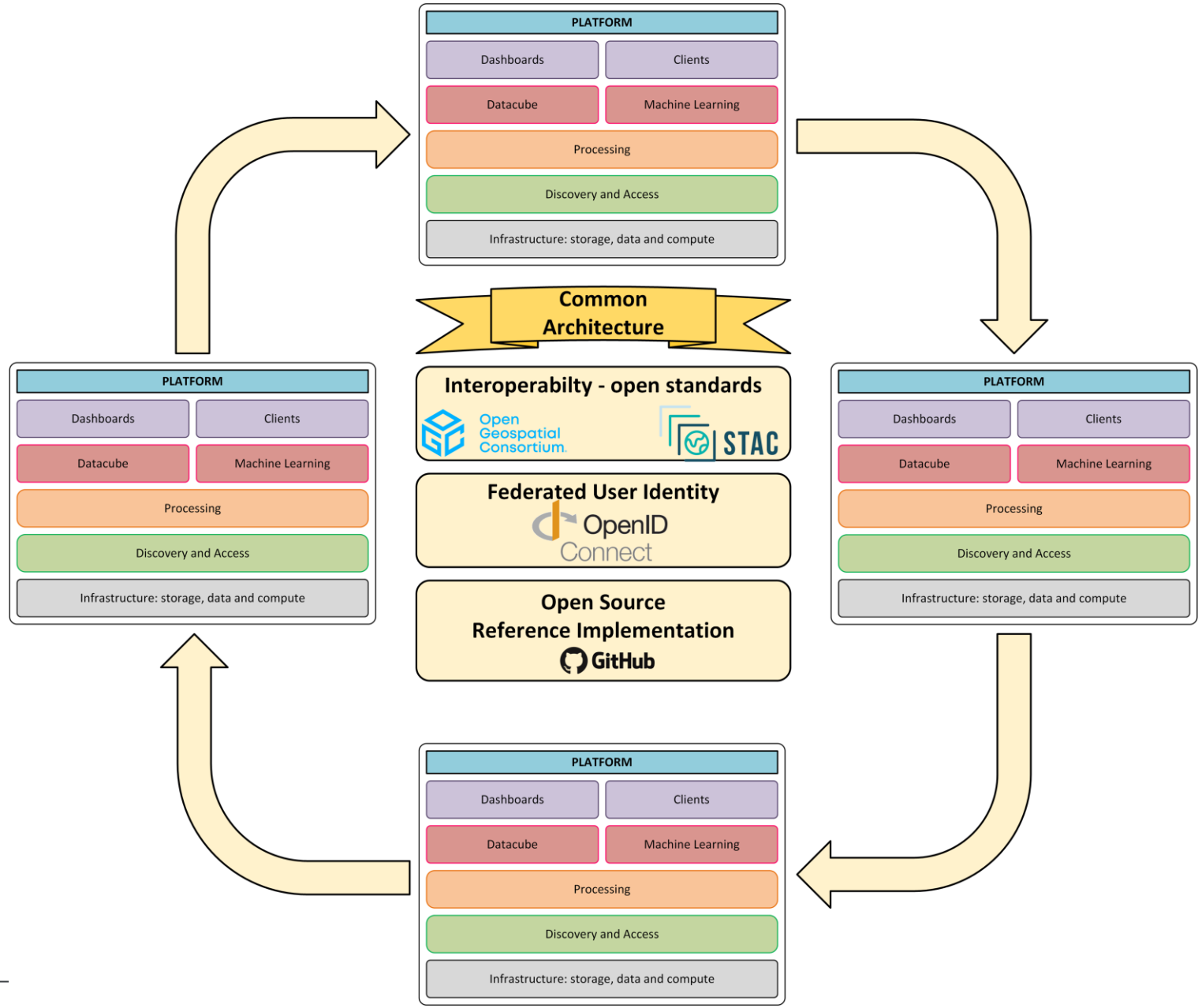
Common Architecture

EOEPCA
EARTH OBSERVATION EXPOITATION PLATFORMS
COMMON ARCHITECTURE

The goal of the Common Architecture is to define and agree a **re-usable exploitation platform architecture** by identifying a set of common building blocks that provide their services through open interfaces

To encourage federation of EPs through an open consensus-based architecture for EPs in the Network of Resources

To provide an **open-source Reference Implementation** of the architecture



Architecture

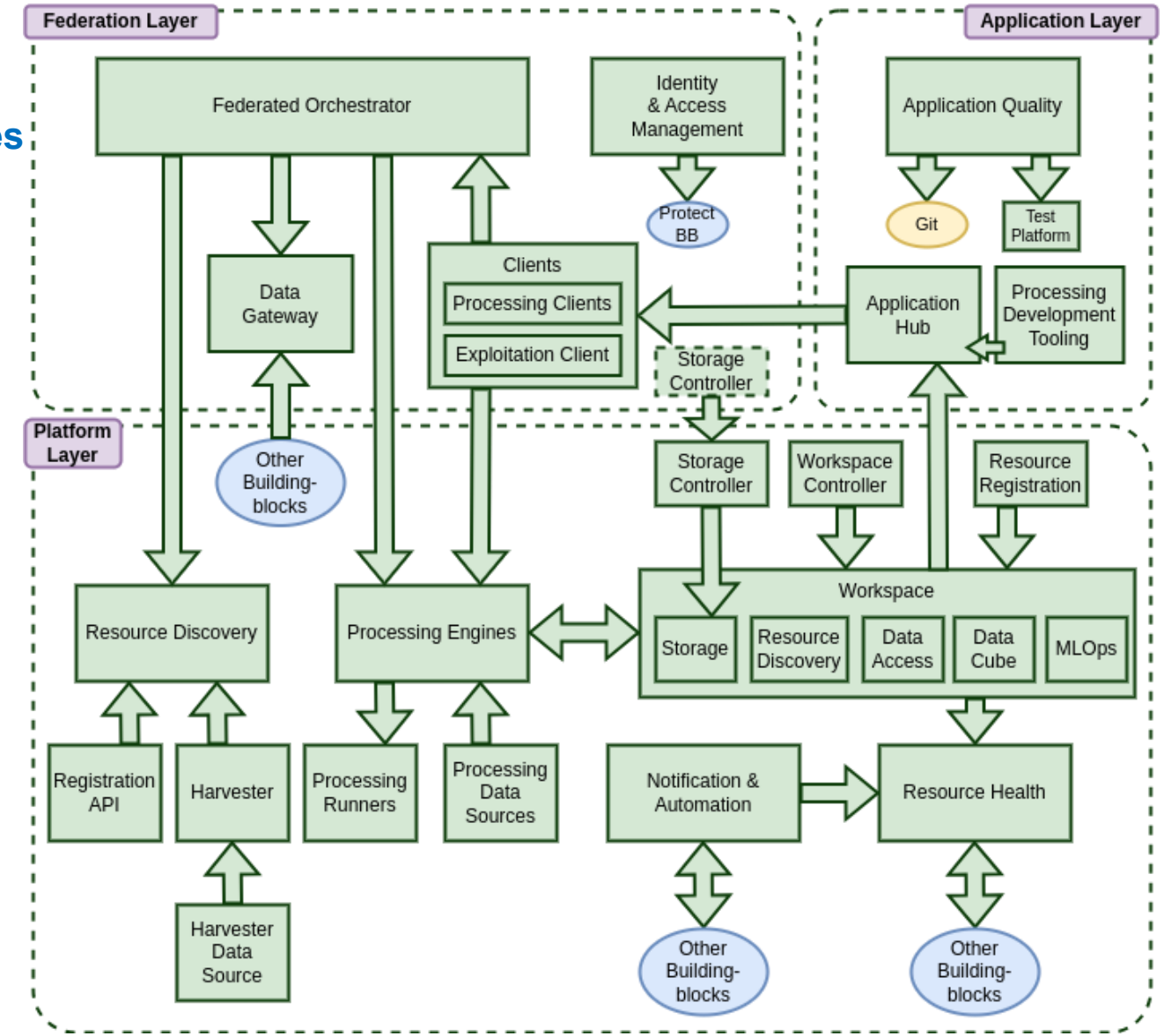
Reference Implementation of the Common Architecture defined by Building Blocks with open standard interfaces

What is a Building Block (BB)?

- Software component that implements a specific platform capability
- Typically provides a service interface (REST API) -> standards
- Dedicated helm chart for each BB – for Kubernetes deployment
- Designed to be used on their own, or in combination as a system
- Open Source

Community Oriented

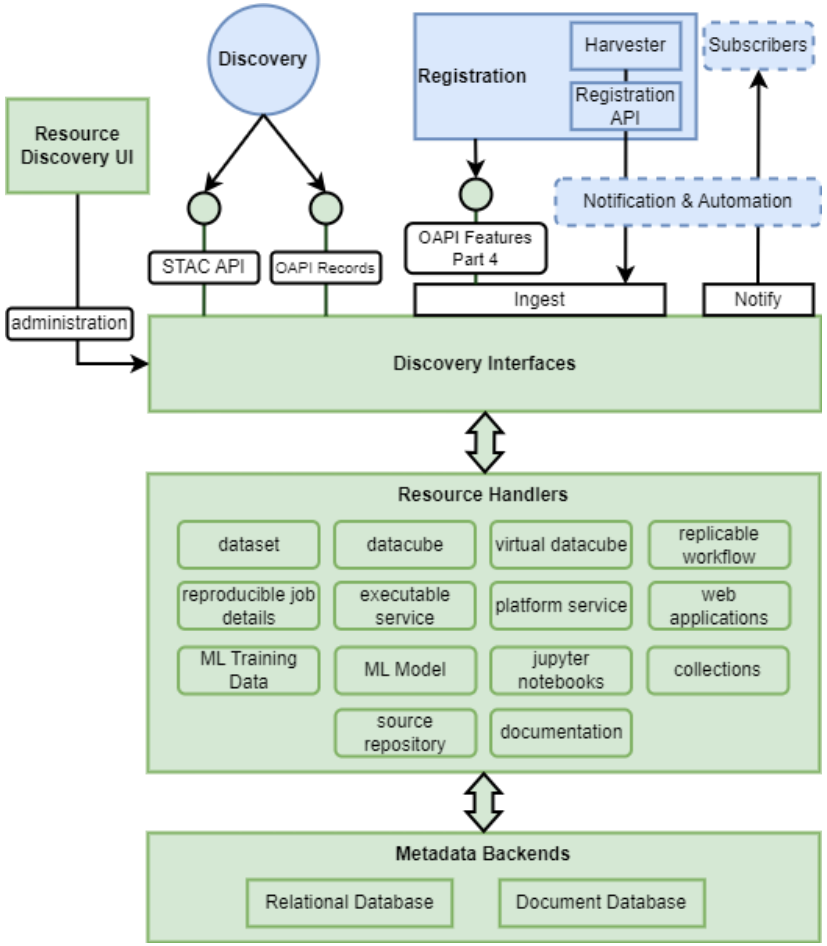
- Open invitation to engage
  - Use cases and Requirements Definition
  - Co-design and Co-development
  - Adopter
- OSGeo Community project
- OGC – Working Group participation and Testbeds



Resource Discovery & Data Access

elasticsearch | stac-fastapi STAC | pgstac | resources pycsw | datasets eo API | Tiler | TiPg

Metadata Catalogue for the resources held within a platform



Standard APIs – STAC and OGC

Shared database for discovery and access

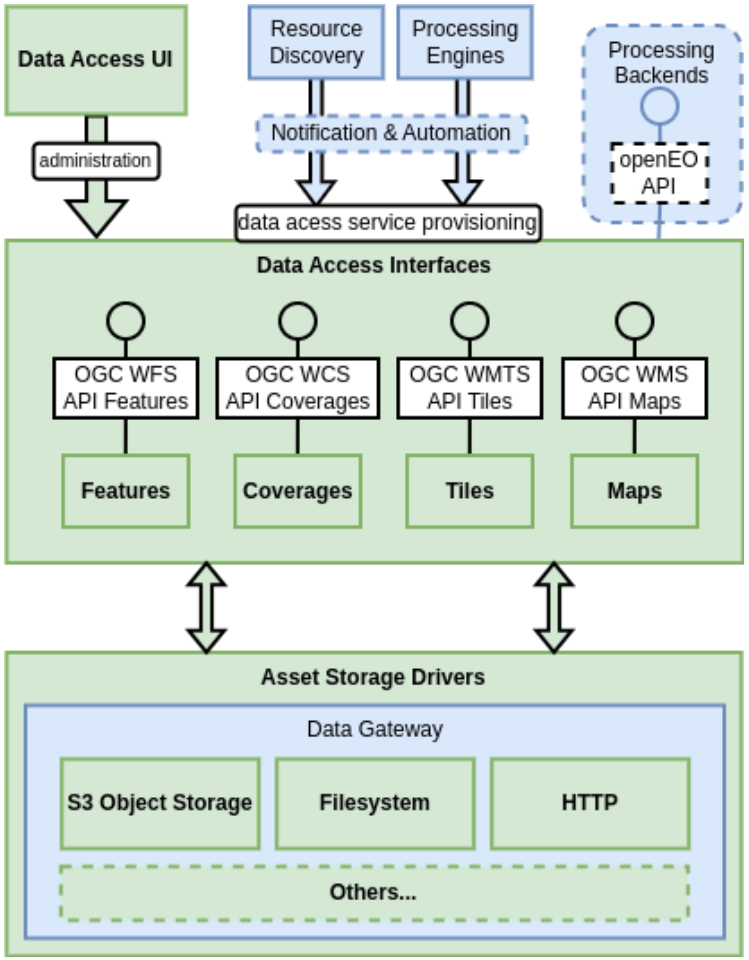
OGC Pub/Sub for notification of new/updated data

Reproducible Open Science – new resource types

Web UI for administration of records and data

Modular design for extensibility: resource types and data sources

Reusable services for data retrieval and visualisation



Processing

Hosted execution of processing workflows

*Supporting **OGC API Processes** and **openEO***

Processing Engines

Service API through which processing workflows are submitted for execution.

Processing Runners

Execution environment for the processing execution.

Processing Data Sources

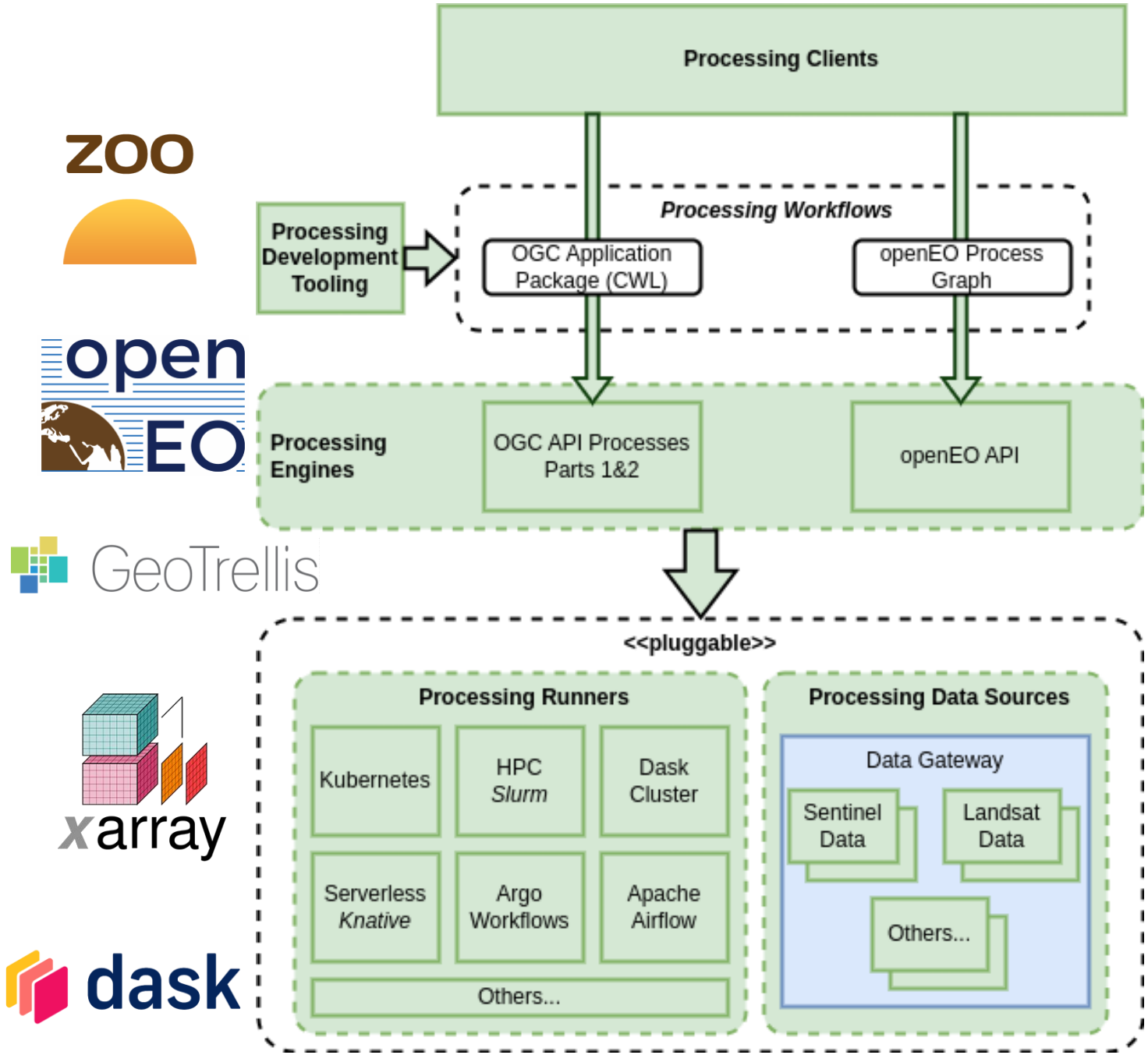
Integration with a data sources to make the dataset available as a processing workflow input

Processing Client

Programmatic access to the processing engine services via its API.

Development Tooling

Web-enabled tooling to aid creation of processing workflows and use of the service API.



Application Package – OGC Best Practice

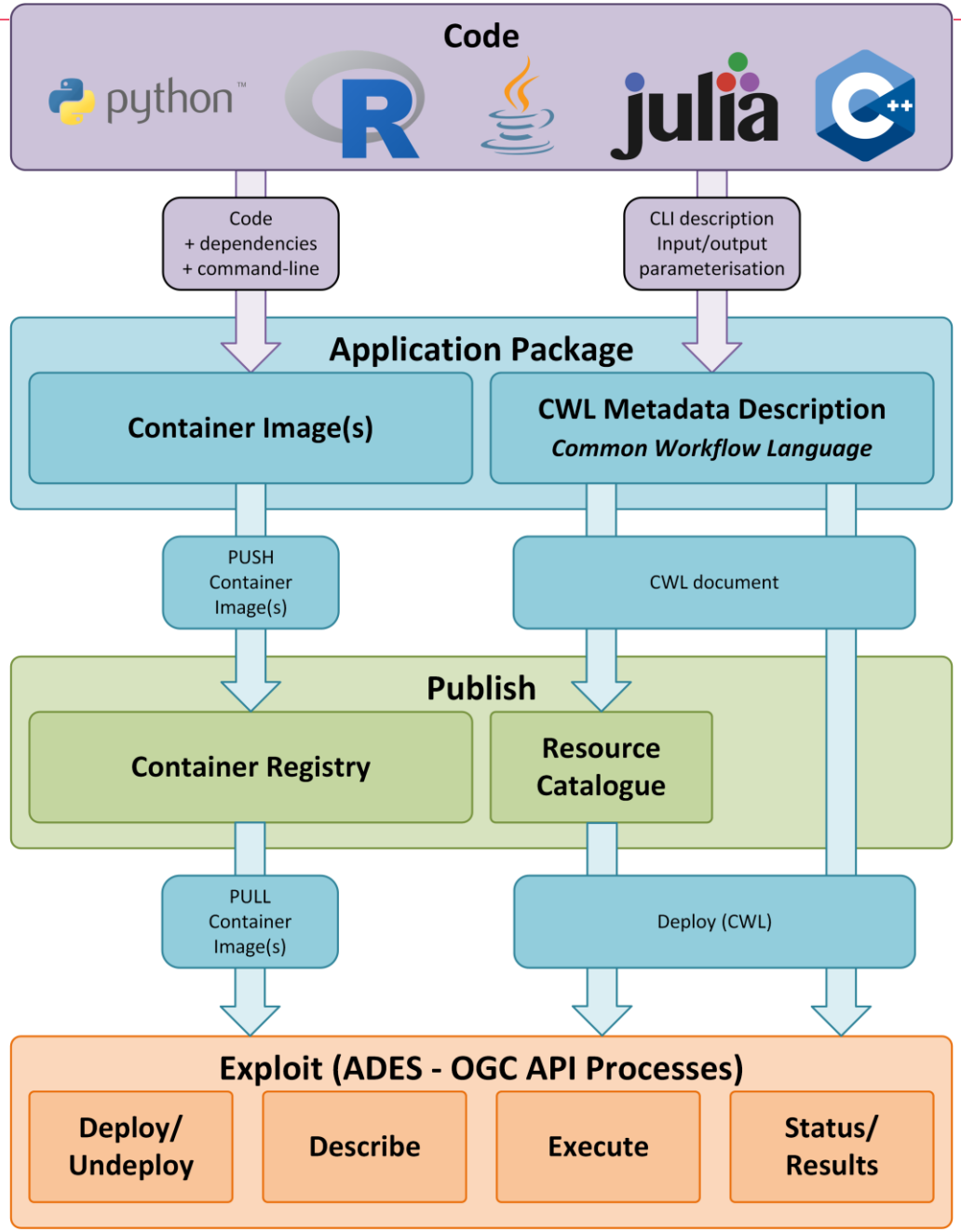
Application Package – Portable User-defined Processing
"A platform independent and self-contained representation of an Application, providing executables, metadata and dependencies such that it can be deployed to and executed within an Exploitation Platform"
…OGC Best Practise for EO Application Packages

Portable Applications
CWL (Common Workflow Language), Container Images, STAC

Container Image(s) for application 'steps'
Application code + dependencies with a command-line entry-point
Language independent and self-contained

CWL Document
Describes the steps (one or more) of the processing (workflow)
Describes input/output parameterisation
Supports various approaches to parallelise steps – Scatter Patterns
Deploy to ADES for processing (OGC API Processes)

OGC Best Practise for EO Application Packages
Provides profile of CWL for application packages
Describes use of STAC for data stage-in/out



Workspace

A collection of services for management of 'owned' resources that are scoped according to global (platform), project/group and user ownership

Workspace Controller

API for administration of workspaces

Declarative provisioning

vCluster for each workspace

Storage Controller

API for self-service management of storage buckets.

Delivery of object storage assets via direct HTTP

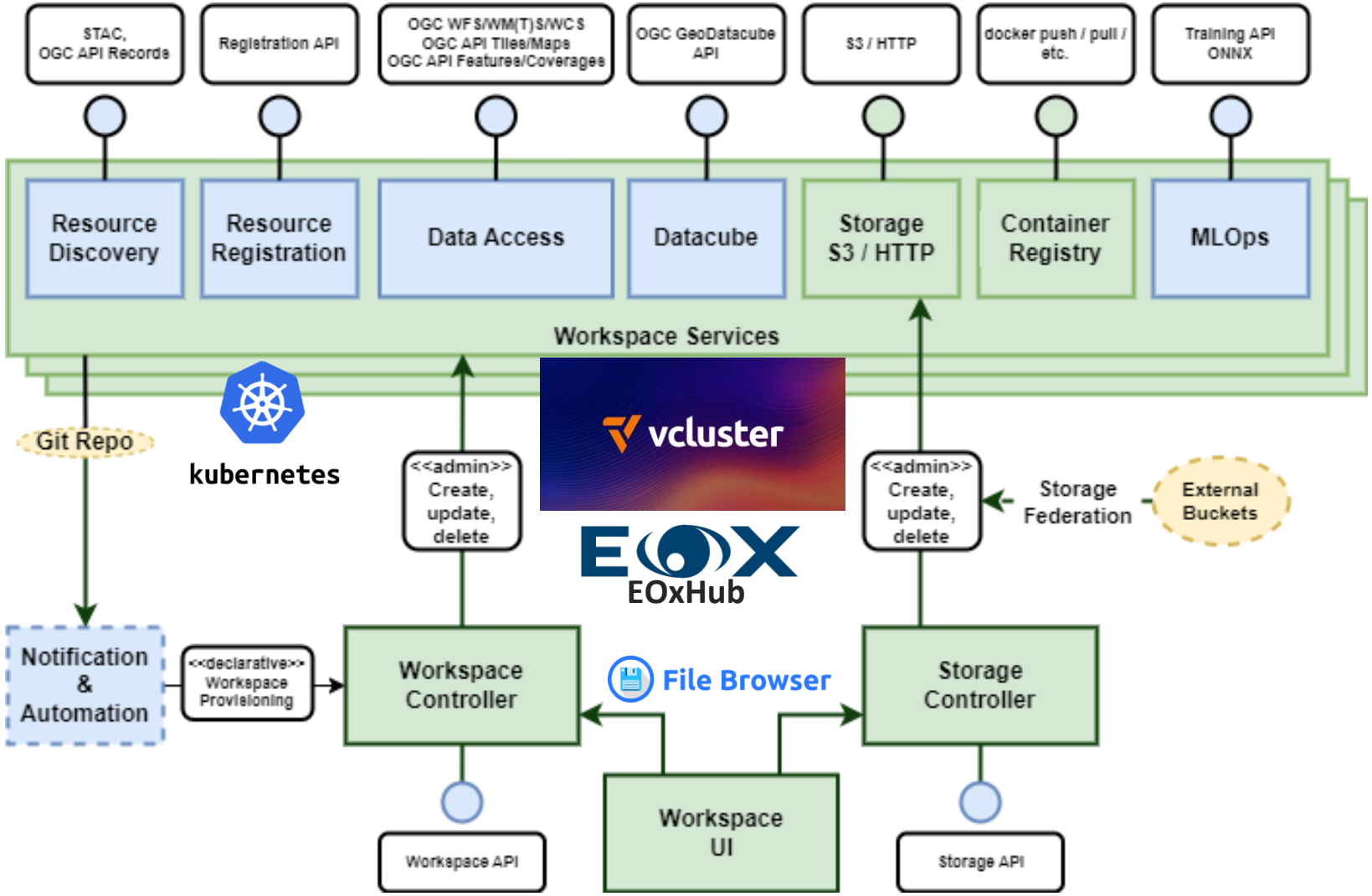
Workspace Services

Extensible set of exploitation services

Comprising instances of other BBs

Workspace UI

Web-enabled UI for interactive usage of workspace and storage capabilities

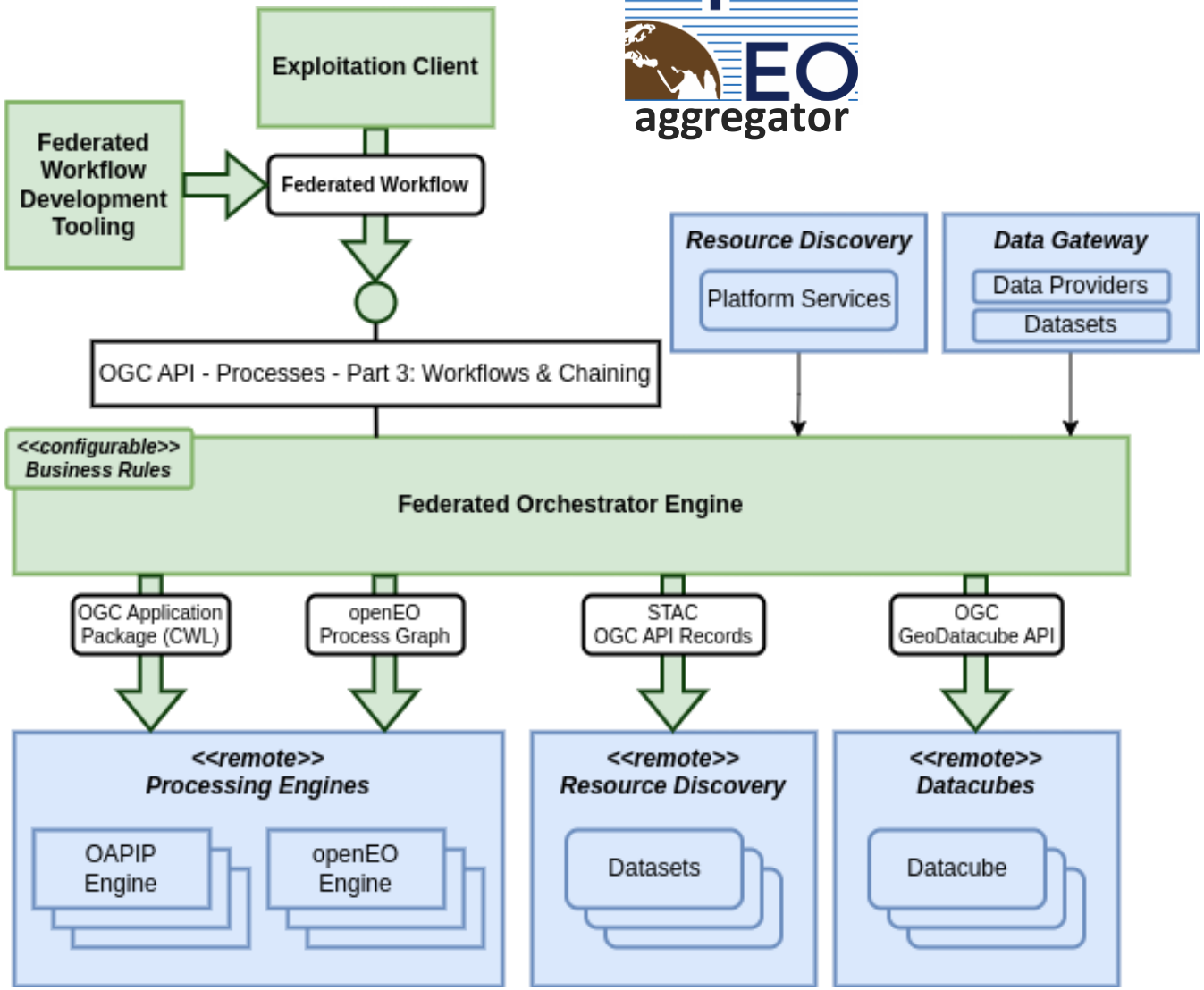


Federated Orchestrator

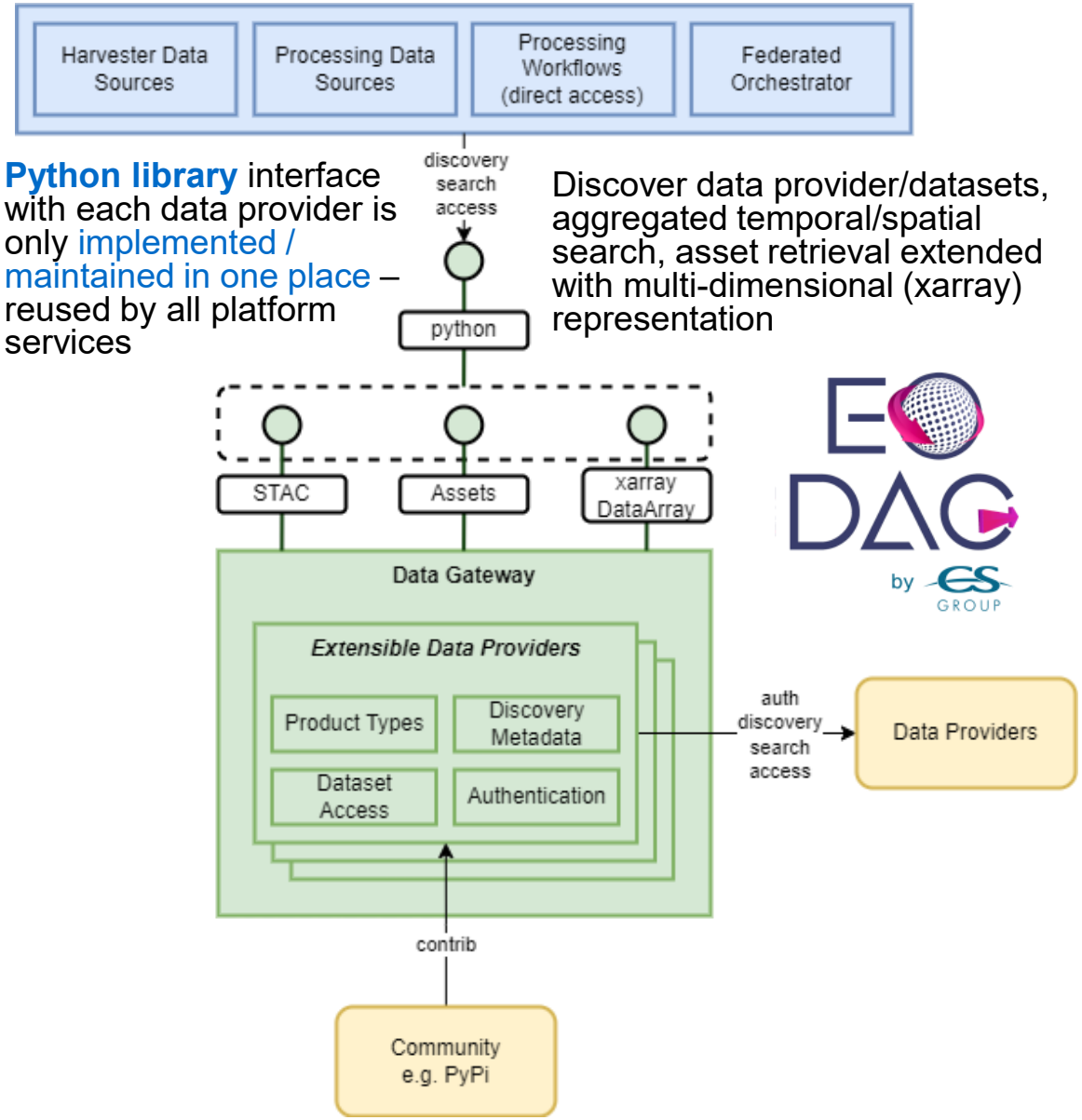
Processing workflow execution across platforms, and across workflow types.

New architecture building block

To avoid the limitation of scientific workflows being constrained within the offering of one or other technology variant, and so create the opportunity to reuse and combine existing published processing workflows of any variant



Data Gateway and Datacube Access

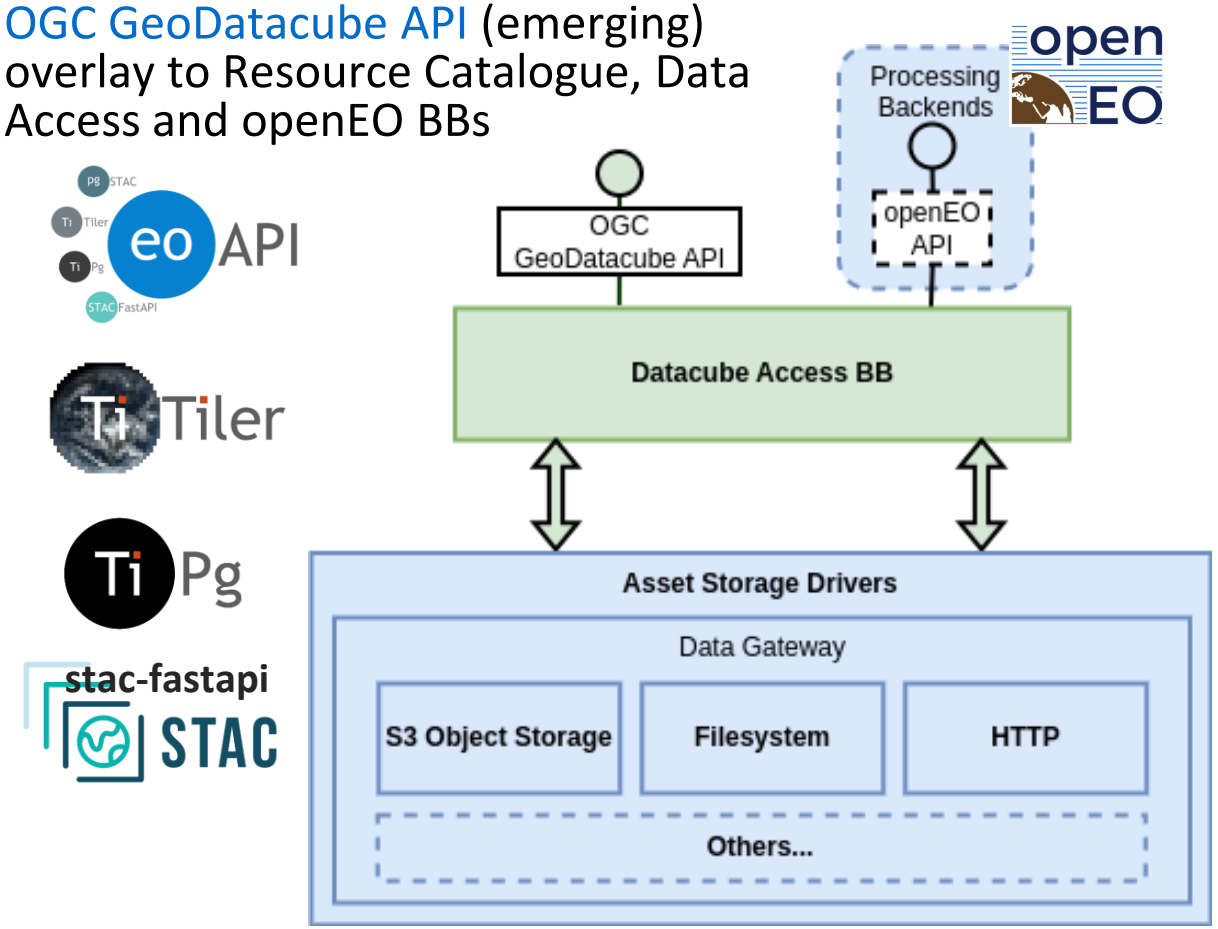


Python library interface with each data provider is only implemented / maintained in one place – reused by all platform services

Discover data provider/datasets, aggregated temporal/spatial search, asset retrieval extended with multi-dimensional (xarray) representation

Datacube Access to harmonise access to multidimensional data – and in doing so facilitate fusion of data from multiple sources by allowing alignment of units, geometries, references systems, etc

OGC GeoDatacube API (emerging) overlay to Resource Catalogue, Data Access and openEO BBs



Resource Registration

Ingestion of resources and their associated metadata into the platform services
e.g. catalogue, data access

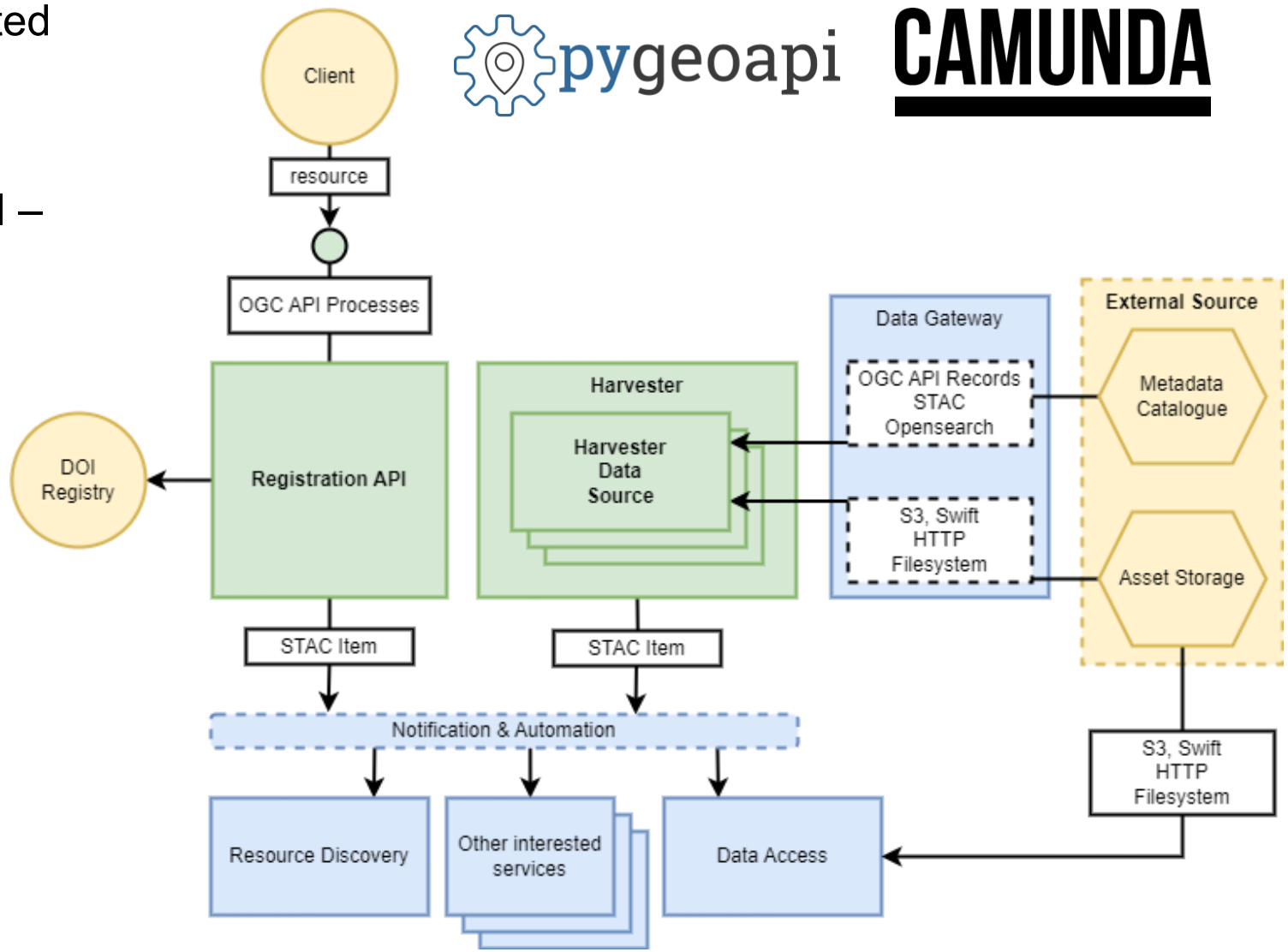
OGC API Processes for Registration API – extensible for different resource types

Harvesting from external catalogues via standard interfaces, using Camunda workflows for large scale data ingestion

Modular design:

- Registration API
- Harvester
- Harvester Data Sources

Registration of DOI for resources
Publish/subscribe for BB decoupling



MLOps

Support services for training of machine learning models

SharingHub

- Management of training data, training runs and model artefacts
- STAC search / representation of datasets and models

Model

- Support popular model frameworks - TensorFlow, PyTorch, etc.
- Standard model representation – e.g. ONNX

Training Runs

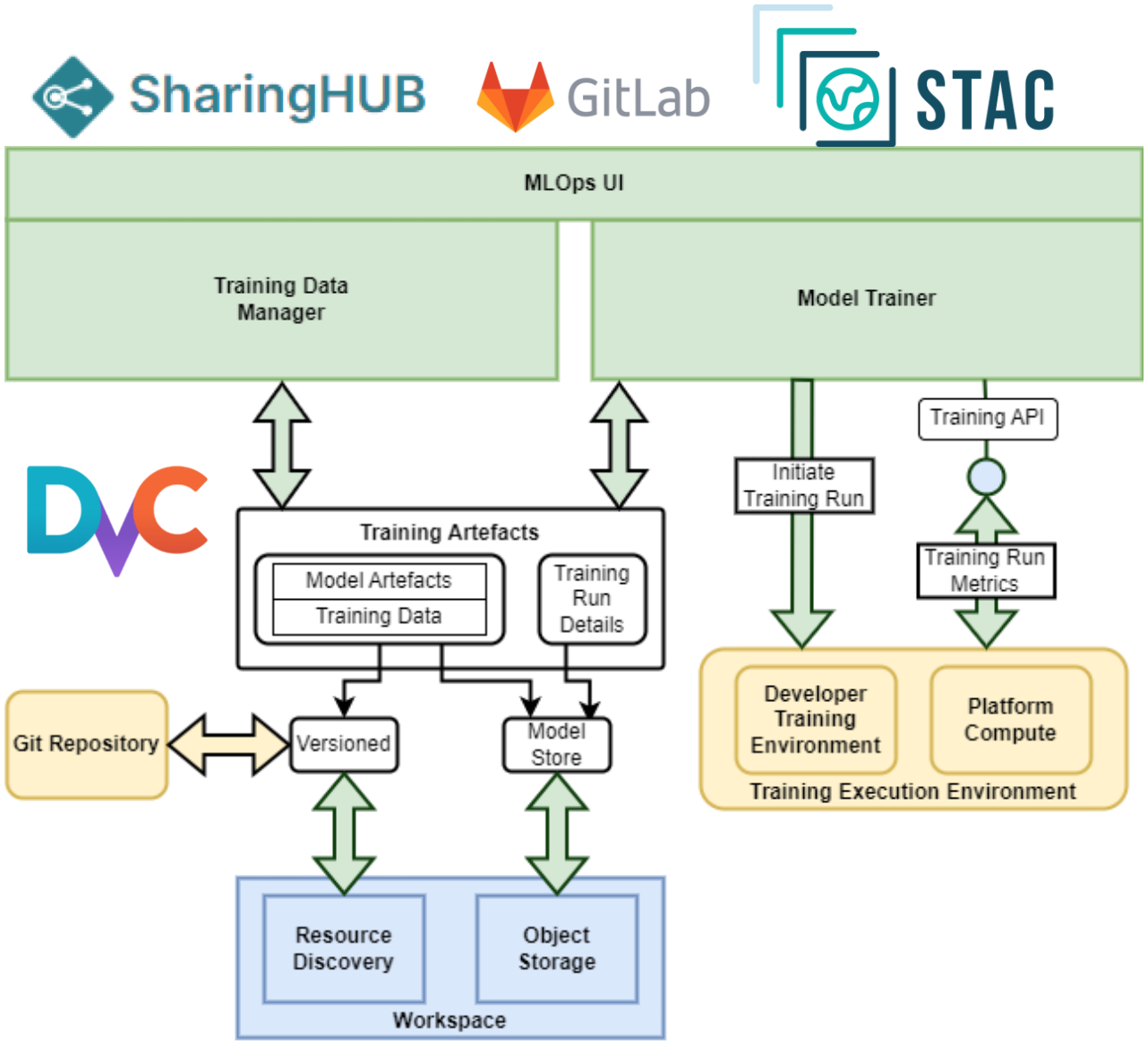
- Manage model artefacts
- Collect run metrics / outputs via API
- Maintain detailed history of runs
- Web UI to assess runs and visualise history

Training Data

- Version controlled via DVC

Persistence

- GitLab for persistence of model runs (version controlled)
- Publish versioned models via Workspace / Resource Discovery



Application Hub

Hosting web-enabled interactive applications – exposed for public consumption.

User extensible set of applications

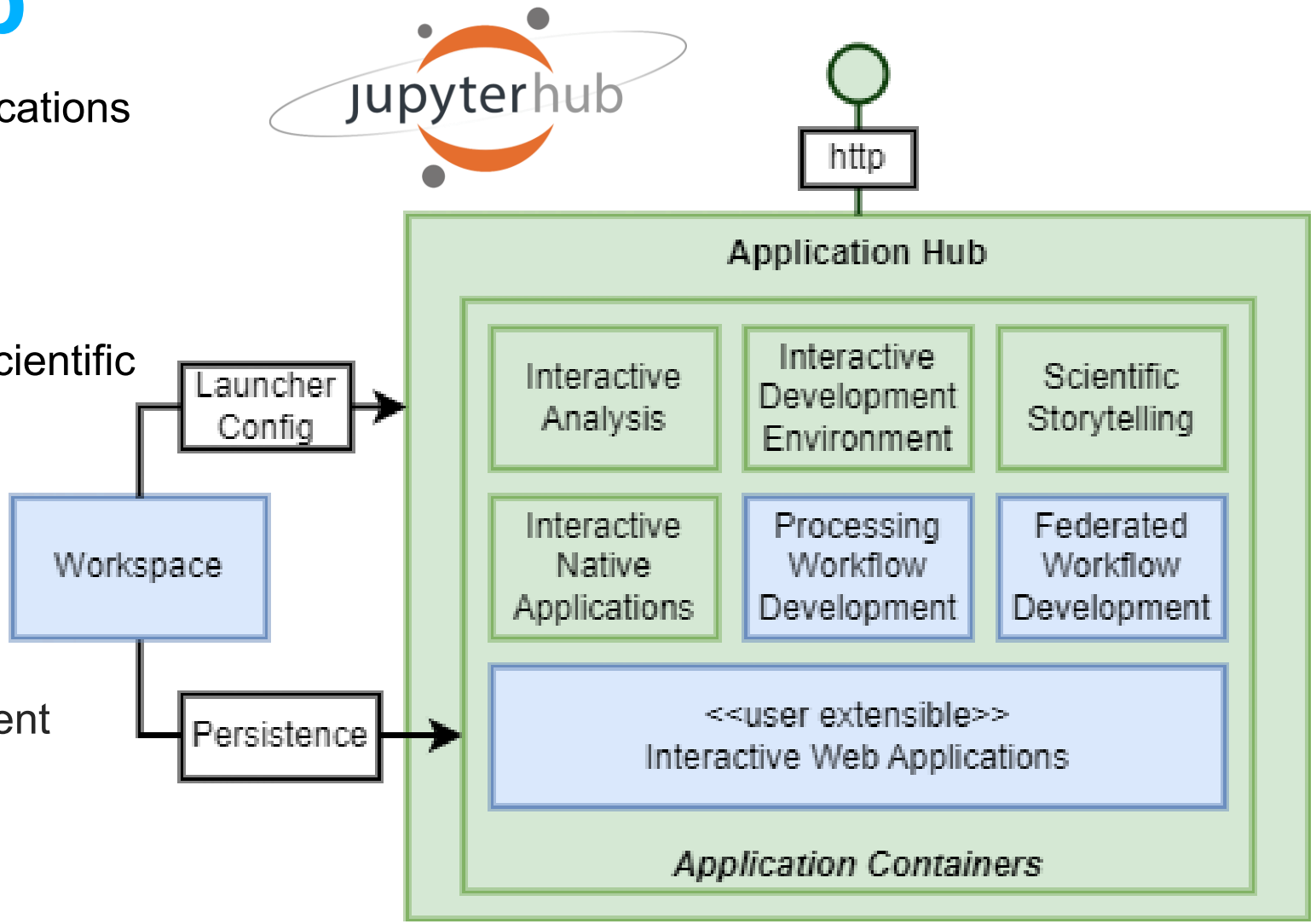
Workspace integration to showcase scientific research…

- Storytelling
- Dashboards

Components:

- Interactive Analysis
- Interactive Development Environment
- Scientific Storytelling
- Interactive Native Applications

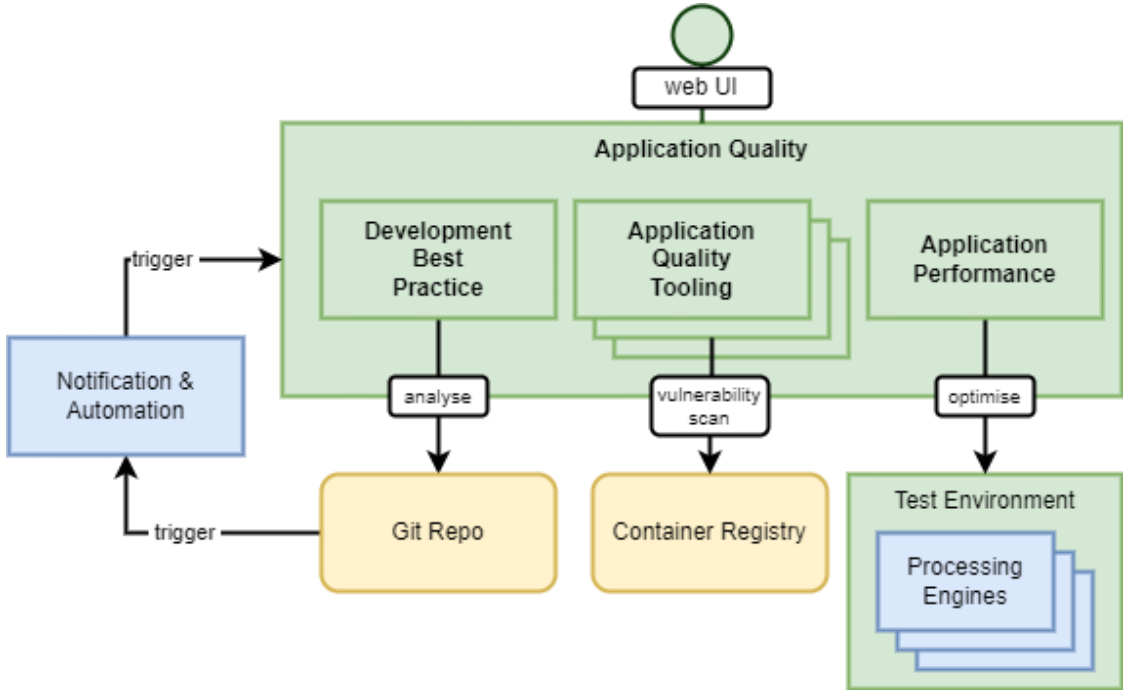
Enhanced development tooling
Storytelling capabilities



Application Quality and Resource Health

Application Quality

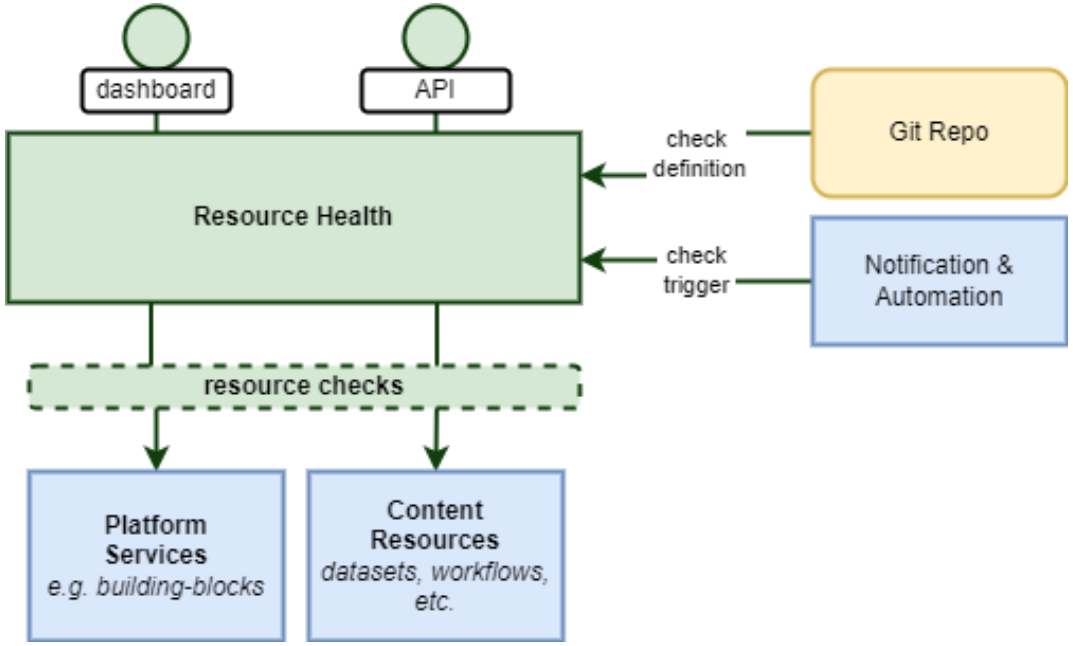
Tools for developers of processing workflows to improve the quality of their software by verifying non-functional requirements and encouraging best practice



Resource Health

Supporting the platform operators and users to monitor the health of the platform resources for which they are responsible

E.g. published datasets or processing workflows



Grafana sonarqube APACHE JMeter™ pytest LOCUST Gatling

Identity & Access Management

User identity (authentication) and access management (request authorisation).

Standard OpenID Connect interfaces.

Identify federation through external providers.

Two goals:

- Provide an out-of-the-box solution
- Support integration of BBs with existing platform IAM solutions

Platform integration

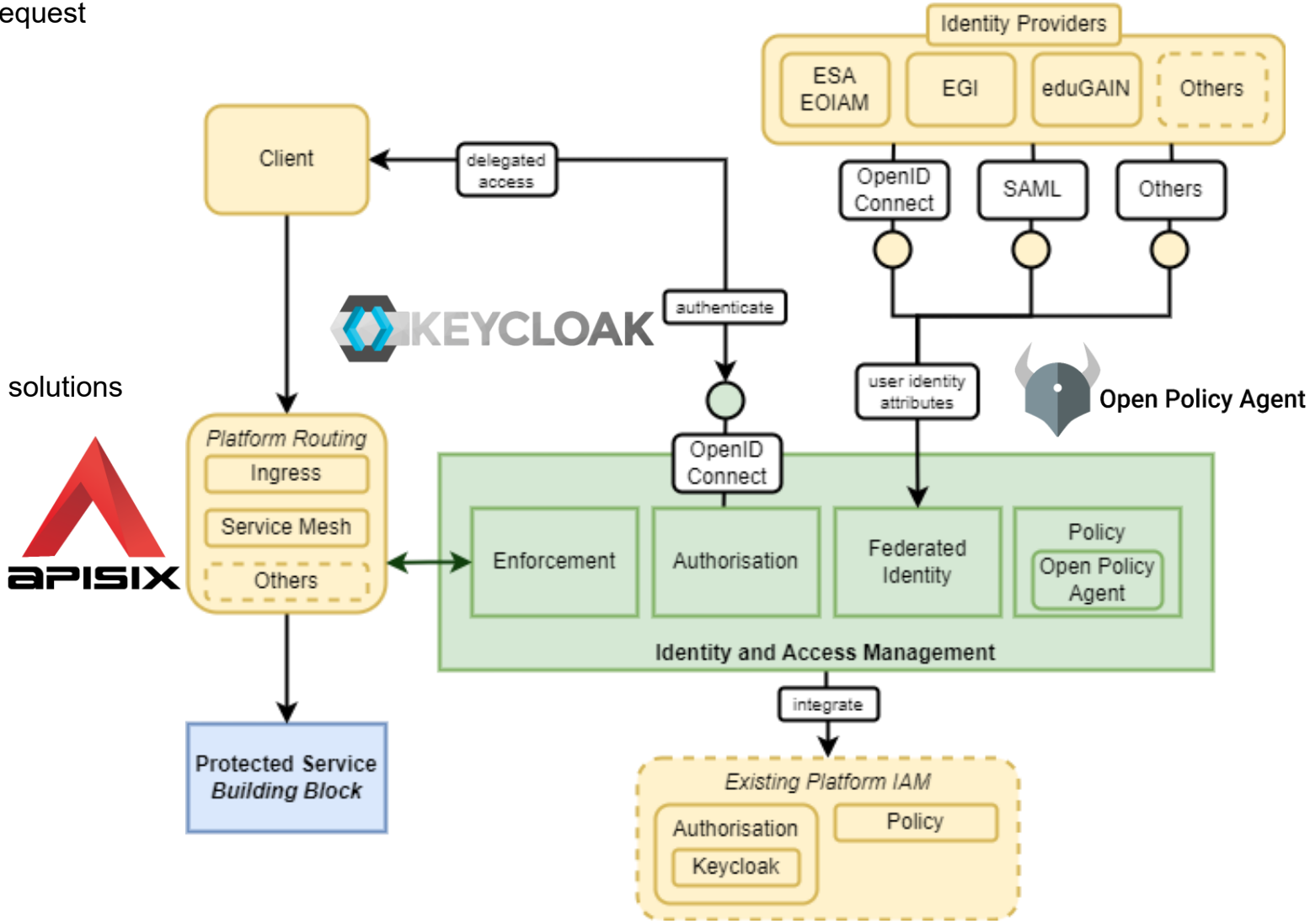
- Existing Authorisation and Policy solutions
- Existing service and request routing patterns

Components:

- Authorisation
- Federated Identity
- Policy
- Enforcement

Standard approach to policy

Focus on integration – platform / identity



EOEPCA Playground

https://killercoda.com/eoepca

Follow a step-by-step deployment of each Building Block directly in the browser

EOEPCA Resource Discovery BB

Check Prerequisites

As usual for EOEPCA, we will use the EOEPCA Deployment Guide scripts to help us configuring and deploying our application.

First, we download and uncompress the **eoepca-2.0-rc1b** version of the EOEPCA Deployment Guide, to which this tutorial refers:

```
curl -L https://github.com/EOEPCA/deployment-guide/tarba
```

The Rescource Discovery deployment scripts are available in the `resource-discovery` directory:

```
cd deployment-guide/scripts/resource-discovery
```

Now we need to understand our pre-requisites. In general EOEPCA Building Blocks will require as minimal pre-requisite a Kubernetes cluster, with an ingress controller to expose the EOEPCA building block interfaces and DNS entries to map the EOEPCA interface endpoints. In this tutorial, for simplicity, nginx is already installed and the `*.eoepca.local` domain is

Editor Tab 1 + 58 min

```
controlplane:~/deployment-guide/scripts/resource-discovery$ bash check-prerequisites.sh

Earth Observation Exploitation Platform Common Architecture Deployment Guide scripts

These scripts accompany the EOEPCA Deployment Guide and are used to configure the deployment of the EOEPCA Building Blocks.
https://eoepca.readthedocs.io/projects/deploy/en/latest/

State variables are stored in /root/.eoepca/state.
As you are running this script for the first time, you will be prompted to configure some settings. These settings will be saved for future run
s and ensure integration between the Building Blocks.
You can use enter to accept the default values or previously configured values.

Please configure the following settings:

Specify the HTTP scheme for the EOEPCA services (http/https) [Default: https]: http
Specify the Ingress class for the EOEPCA services (apisix/nginx) [Default: apisix]: nginx
Enter the base domain name [Default: example.com]:
```

Learn the basic usage instructions of how to use that Building Block.

Add STAC Collections and Items

So, now we have our Resource Discovery catalogue, we need to fill it with products.

To do so, we can use the catalogue STAC APIs, for which you will see details in the Swagger documentation in the same instance you just installed.

Add a collection

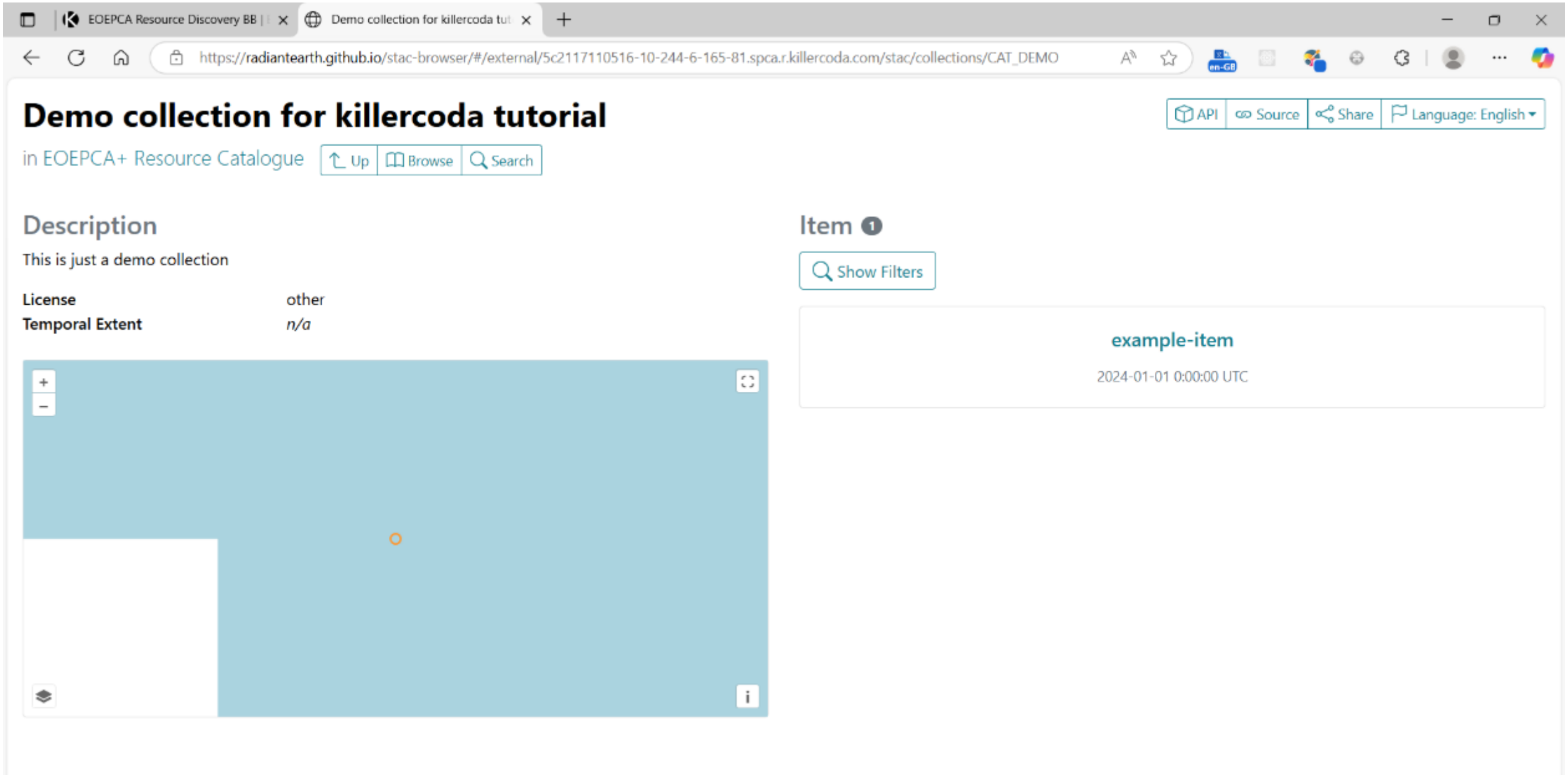
First, we can add a Collection for our data. Let's save first the following STAC

```
cat <<EOF | tee CAT_DEMO.json | jq
{
    "stac_version": "1.0.0",
    "type": "Collection",
    "license": "Open Data",
    "id": "CAT_DEMO",
    "title": "Demo collection for killercoda tutori
    "description": "This is just a demo collection"
    "links": [],
    "extent": { "spatial": { "bbox": [[-180.0, -90.
}
```

Editor Tab 1 +

```
controlplane:~/deployment-guide/scripts/resource-discovery$ while [[ `curl -s -o /dev/null -w "%{http_code}" "http://resource-ca
talogue.eoepca.local/stac"` != 200 ]]; do sleep 1; done
controlplane:~/deployment-guide/scripts/resource-discovery$ bash validation.sh
✅ 1 pod(s) with label 'io.kompose.service=pycsw' are running.
✅ Deployment 'resource-catalogue-service' is ready.
✅ Service 'resource-catalogue-db' exists.
✅ Service 'resource-catalogue-service' exists.
✅ URL 'http://resource-catalogue.eoepca.local' returned expected HTTP status code 200.
✅ URL 'http://resource-catalogue.eoepca.local/collections' returned expected HTTP status code 200.
✅ PVC 'db-data-resource-catalogue-db-0' is bound.
✅ ConfigMap 'resource-catalogue-db-configmap' exists.
✅ ConfigMap 'resource-catalogue-configmap' exists.

All Resources:

NAME                                                READY   STATUS    RESTARTS       AGE
pod/resource-catalogue-db-0                         1/1     Running   0              2m39s
pod/resource-catalogue-service-7d6cf958f5-8hqm5     1/1     Running   3 (101s ago)   2m39s

NAME                                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/resource-catalogue-db         ClusterIP   10.107.18.34    <none>        5432/TCP   2m39s
service/resource-catalogue-service    ClusterIP   10.100.161.39   <none>        80/TCP     2m39s

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/resource-catalogue-service    1/1     1            1           2m39s

NAME                                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/resource-catalogue-service-7d6cf958f5    1         1         1       2m39s

NAME                                       READY   AGE
statefulset.apps/resource-catalogue-db     1/1     2m39s
controlplane:~/deployment-guide/scripts/resource-discovery$
```

The service is temporarily available over ingress.



EOEPCA+: Partner Organisations

EOEPCA
BETTER ACCESS TO EARTH OBSERVATION

esa

TELESPAZIO
a LEONARDO and THALES company
Telespazio Ibérica

NOVASPACE
Communications
& Outreach

DLR
EOX
EOFARM
development SEED

Resource Discovery
Resource Registration
Data Access
Datacube Access
Workspace

TERRAQUE
vito
GeoLabs
eurac research
Matthias Mohr
Softwareentwicklung

Processing
Application Hub
Federated Orchestrator
Notification & Automation

werum
SOFTWARE & SYSTEMS

IAM

CS
a Sopra Steria company

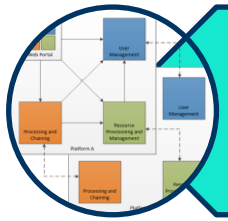
MLOps
Data Gateway

spaceapplications
SERVICES
SENSmetry

Application Quality
Resource Health

Where to find us

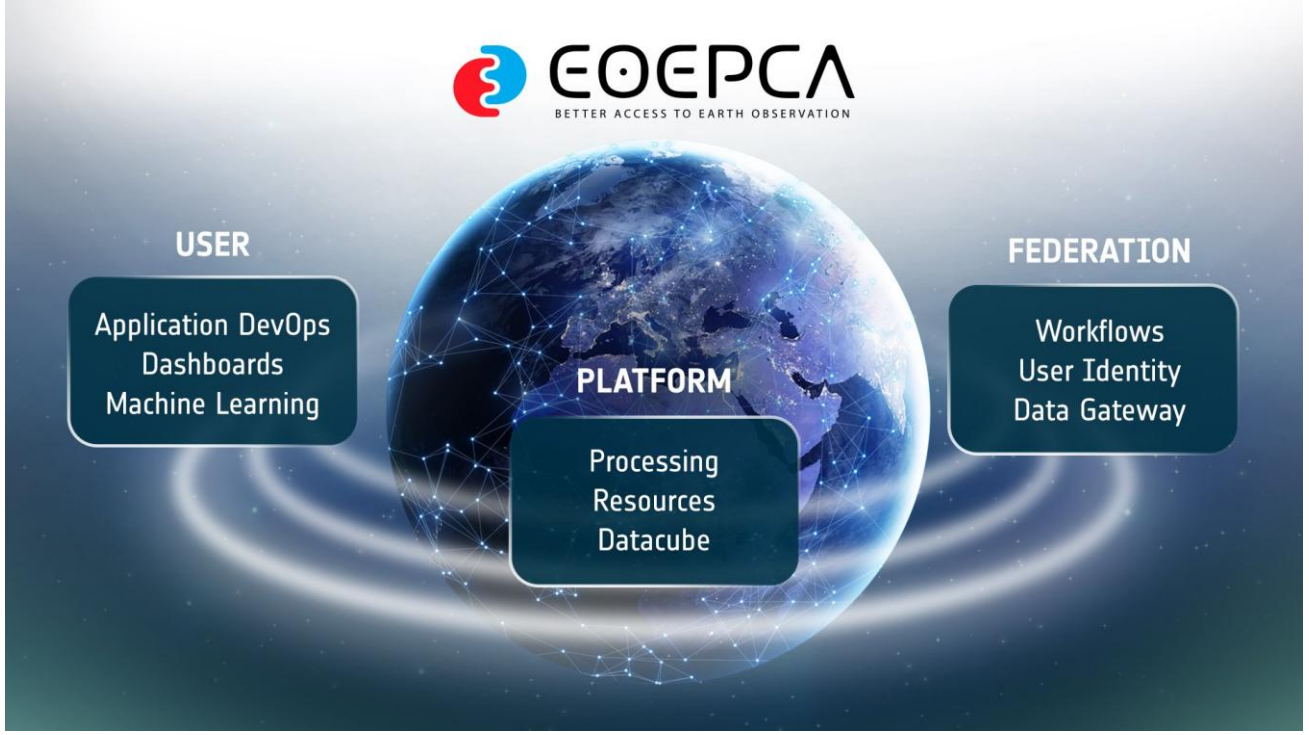
EOEPCA
BETTER ACCESS TO EARTH OBSERVATION



Web Portal
https://eoepca.org/

GitHub
https://github.com/EOEPCA

Documentation
https://eoepca.readthedocs.io/



TELESPAZIO
a LEONARDO and THALES company

esa

THANK **YOU**
FOR YOUR ATTENTION

telespazio.co.uk