EO Application Package Best Practice

Pedro Gonçalves, Terradue

The 120th OGC Member Meeting

14 September 2021

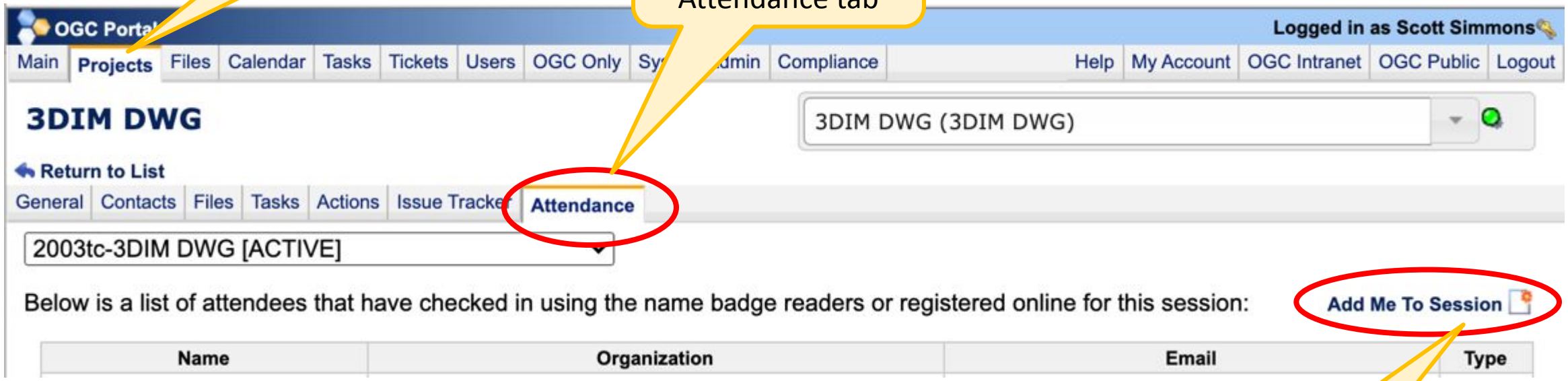
The world's leading and comprehensive community of experts making location information:

- **F**indable
- **A**ccessible
- **I**nteroperable
- **R**eusable

Members: please record your attendance

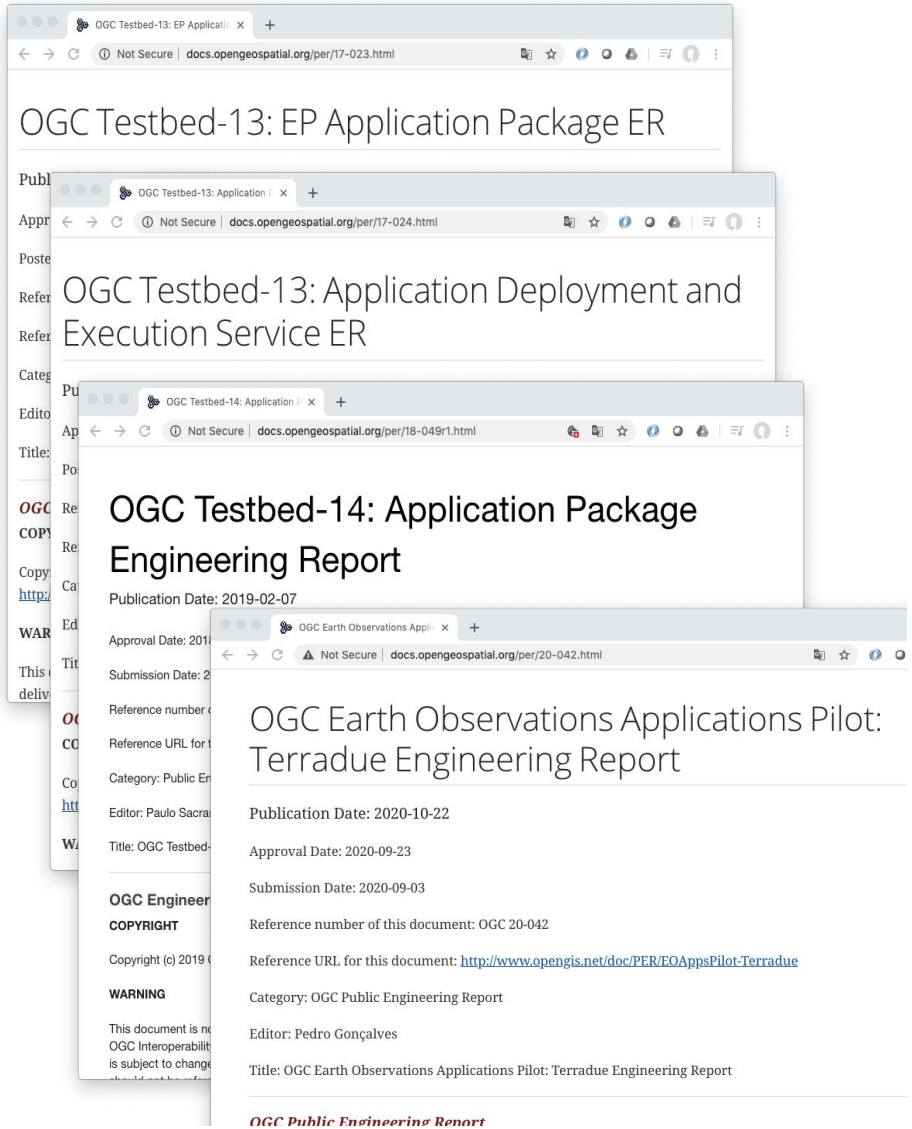
1. Find the appropriate project

2. Go to the Attendance tab



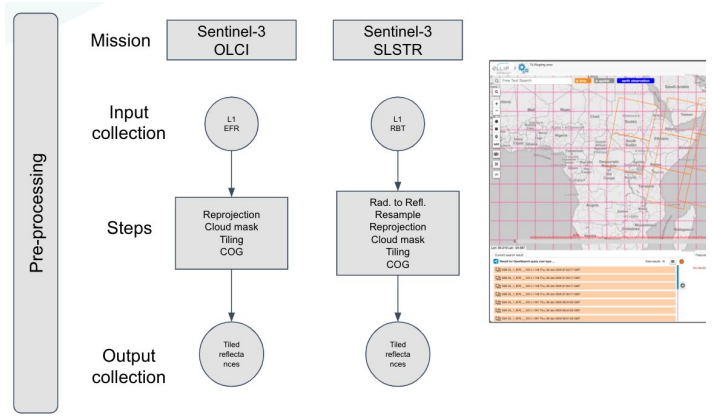
3. Add yourself to the session

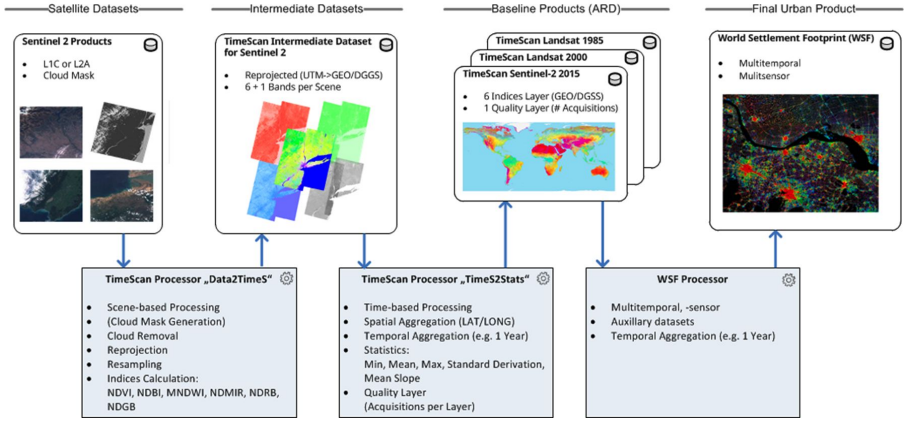
Earth Observation Application Package



- Previous OGC Testbeds 13-16 initiated the design of an application package for Earth Observation Applications in distributed Cloud Platforms
- The application package provides information about the software item, metadata and dependencies
- Deployed and executed within an Exploitation Platform in a service compliant with the OGC API Processes specification

Earth Observation Applications Pilot 2020





- Evaluated the maturity of Application Package in a real world environment with applications brought by several developers that work with data from Earth observation satellites.
- These developers brought different views and requirements in terms of data access and processing to challenge the application package architecture and platform readiness

The role of the Application Package

- Describes the data processing application by providing information about
  - parameters, software item, executable, dependencies and metadata.
- Ensures that the application is fully portable and supports automatic deployment in a Machine - To - Machine (M2M) scenario.
- Application Package information model allows the deployment of the application as OGC API - Processes compliant web service.

High Level Requirements

- Decouple application developers from exploitation platform operators and from application consumers
  - Focus on application development by minimizing platform specific particularities
  - Make their applications available on any number of platforms

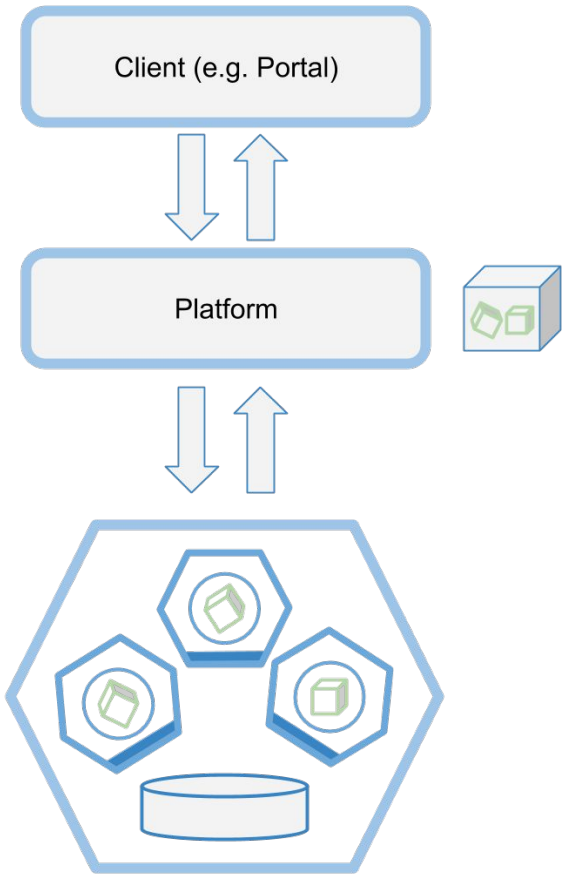
- Enable exploitation platforms to virtually support any type of packaged EO application

EO Application Package

- Application developers create containers with their runtime environment, dependencies and application binaries
- Application developers/integrators describe the Application package using the Common Workflow Language
- Applications needing more complex Data Flow Management can use a local catalogue encoded using STAC as a data manifest for application inputs and outputs metadata
- Application can be deployed in a Cloud provider and invocable in a service compliant with the OGC API Process specification

EO Application Package Best Practices

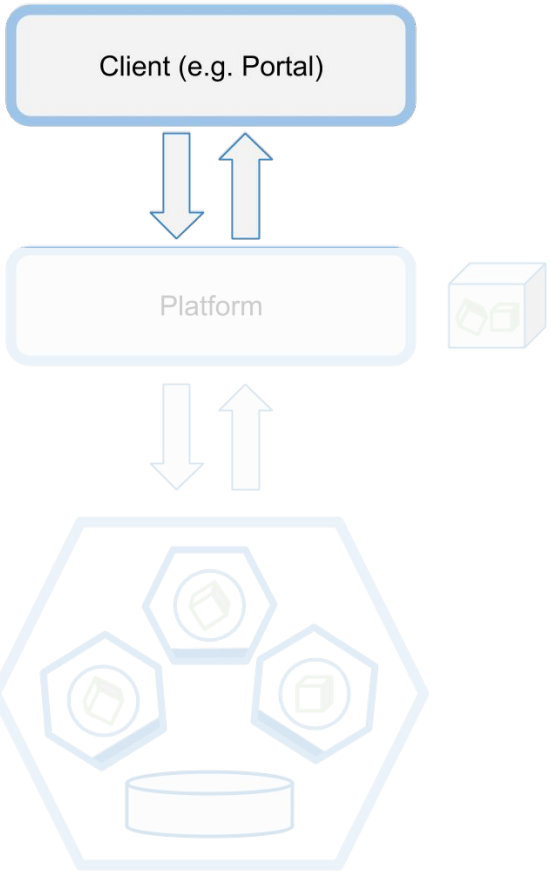
OGC

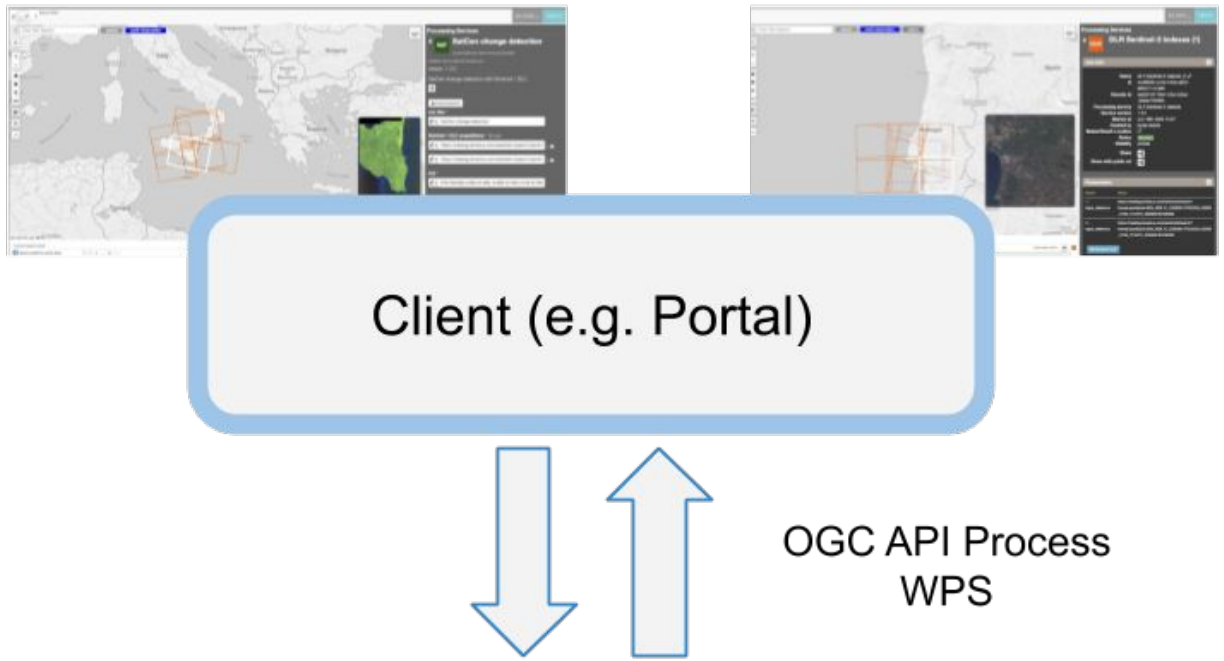


- The application (e.g. Python, R, JAVA, shell script, C++) is containerized and register in Container Registry
- The application package is created in Common Workflow Language
- The application is deployed in an Exploitation Platform
- The service is available for execution with the OGC API Processes specification
- The Platform converts the OGC API Processes requests in a CWL execution request

Service Request from Clients (e.g. Portal)

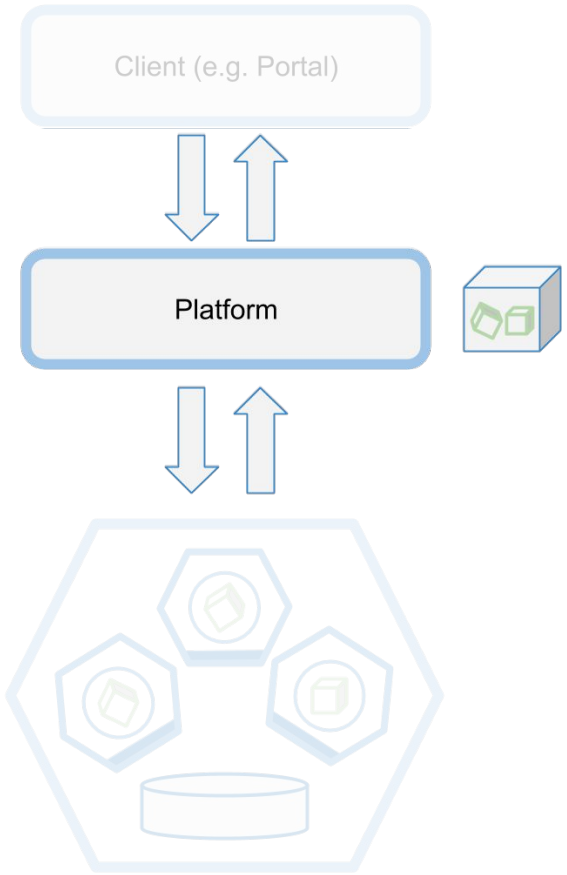
OGC

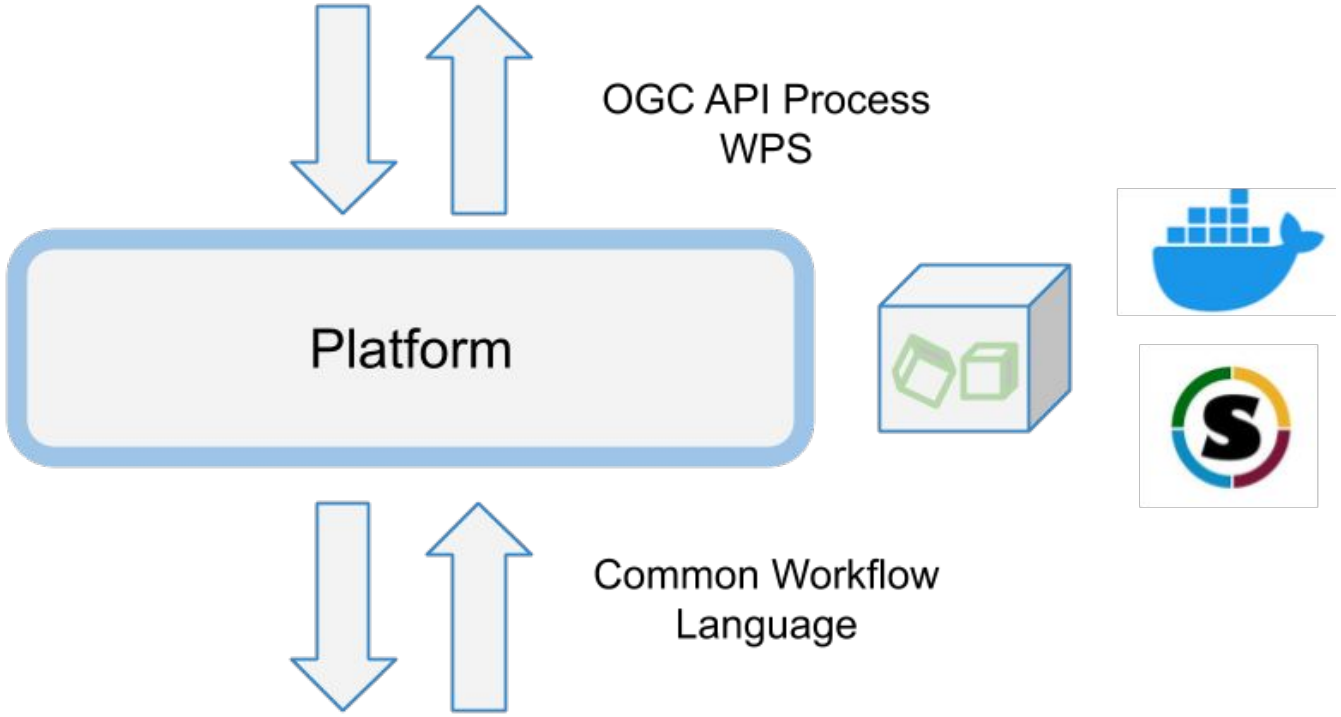




Service Request to Application Execution

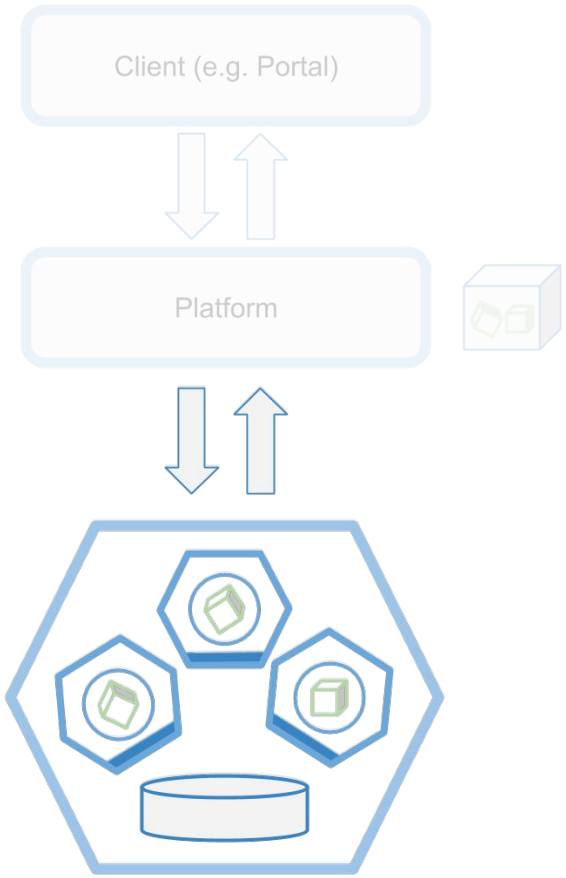
OGC

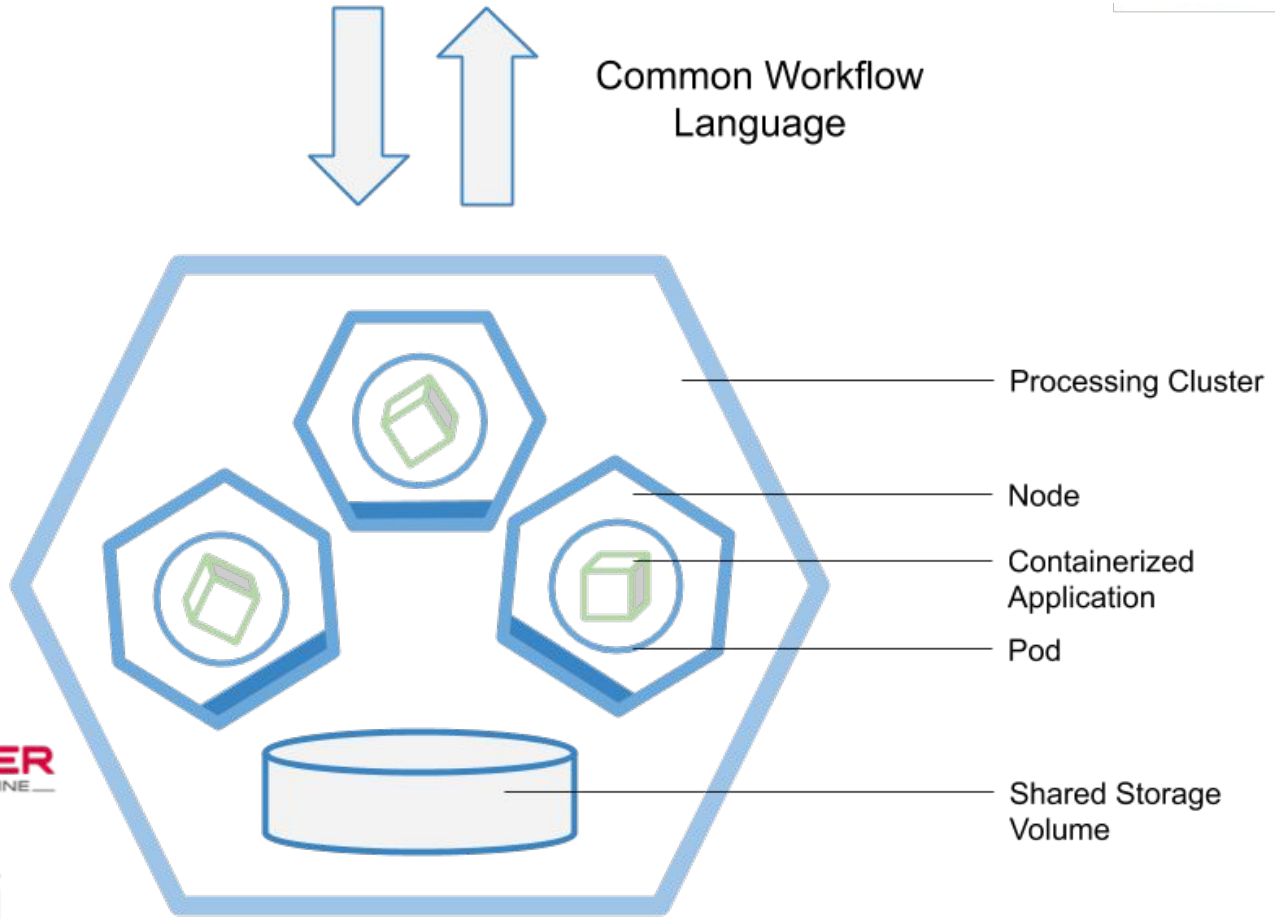




Application Execution

OGC





Viewpoints

OGC

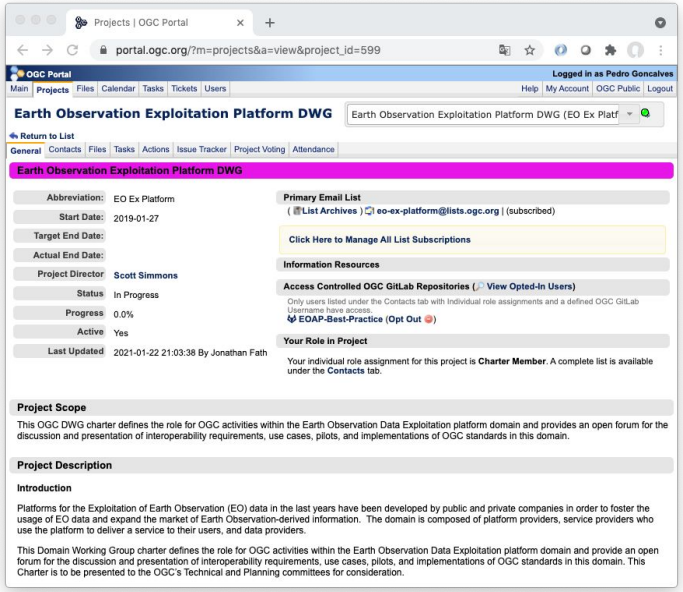
OGC 20-089 defines guidance for the 3 viewpoints:

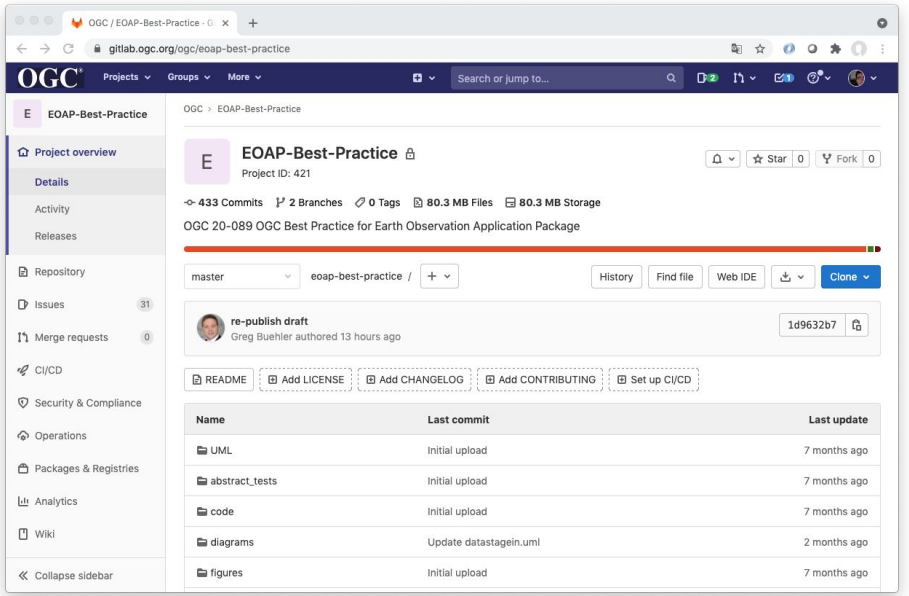
- For a **Developer** to adapt an application (or wrapper)
  - Optional data staging from a local STAC catalogue
  - Create and publish a Docker
- For an **Integrator** to package an application
  - Create the CWL document (command line, inputs, outputs …)
- For an **Platform** to deploy and execute the application
  - Extract the CWL Workflow information
  - Create the OGC API Processes Service Parameters mapped from the CWL
  - Expose the OGC API Processes service

GitLab Access

- Document published in pending documents
  - https://portal.ogc.org/files/?artifact_id=98620
- Document source on OGC GitLab
  - opt-in 1st on the EOEP DWG page

https://gitlab.ogc.org/ogc/eoap-best-practice





Best Practice for EO Application Package

- Hands-on experience from Testbeds and Pilot
- Comments and suggestions tracked and resolved as GitLab Issues
- 11 submitters organisations (let's us know if you want to be added)

Motion to approve an electronic vote for releasing OGC 20-089 as a OGC Best Practice

The EO Exploitation Platforms DWG recommends that the TC approve an electronic vote for releasing OGC 20-089 "Best Practice for EO Application Package" as a OGC Best Practice.

- Pending any final edits and review by OGC staff
-
- Motion: Günther (ESA)
- Second: Ryan (Geo COnnections)
- Discussion:
- There was no objection to unanimous consent

This document defines the Best Practice to package and deploy Earth Observation Applications in an Exploitation Platform.

TERRADUE

Looking forward hearing from you!

https://www.terradue.com

Pedro Gonçalves

pedro@terradue.com

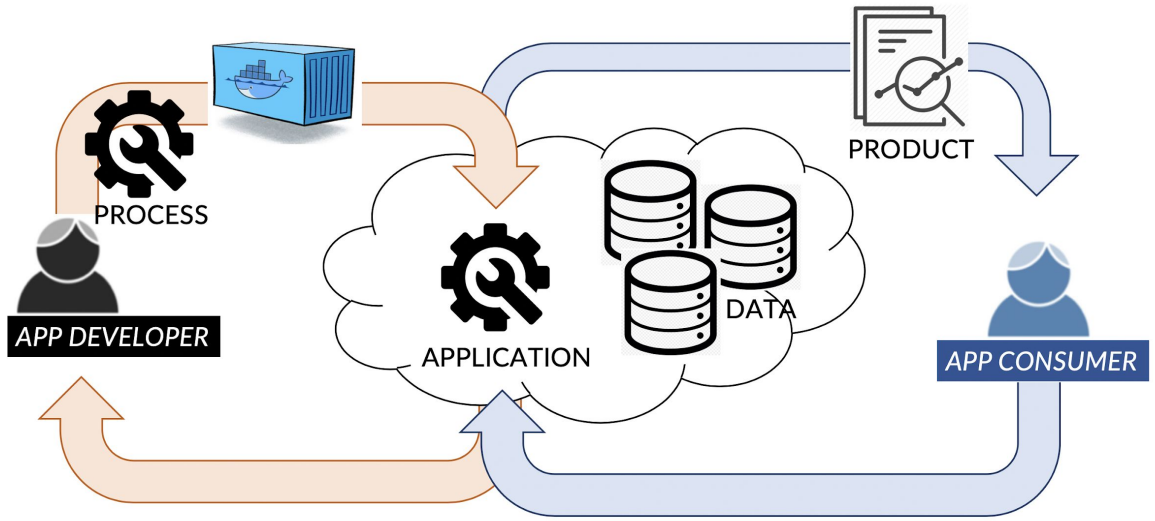
Backup Slides

- Usage Scenarios
- Common Workflow Language
- Data Staging with STAC
- Viewpoints explained

Usage Scenarios

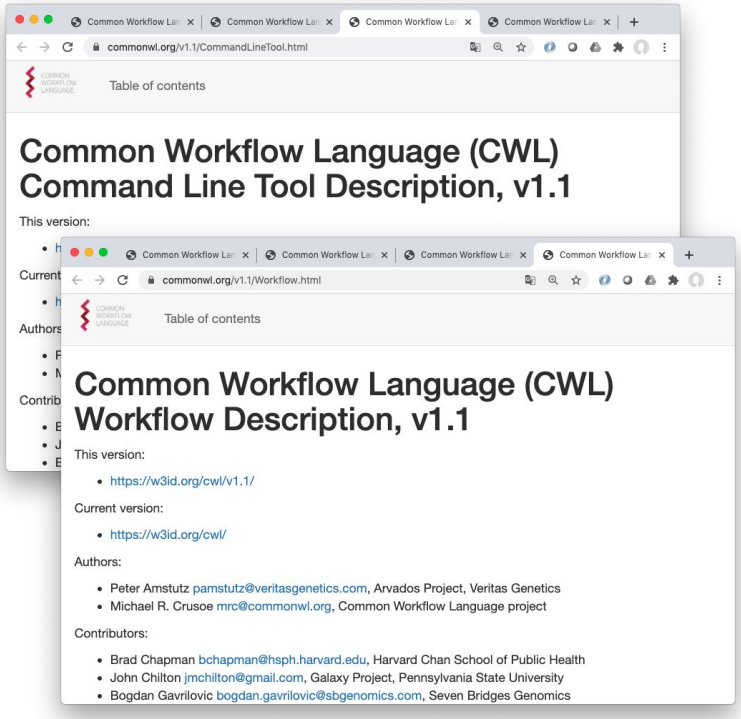
Build to run operations from application testing, validation, and deployment to execution in production

- Alice to package an application
- Bob to script the execution of application
- Eric to deploy an application on platform Z
- Platform Z to accept the deployment of a new process
- Platform Z to execute a process with specific parameters



Common Workflow Language

OGC



- The CWL is a specification for describing analysis workflows and tools
- Portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high performance computing (HPC) environments
- Design to meet the needs of data-intensive science, such as Bioinformatics, Medical Imaging, Astronomy, Physics, and Chemistry

Application Package

OGC

Application developers

- Create containers with their runtime environment and dependencies
- Describe the overall package using the **Common Workflow Language (CWL)**.

```
- class: Workflow
  doc: ACME change detection with Sentinel-1 SLC
  id: acme_cd
  label: ACME change detection
```

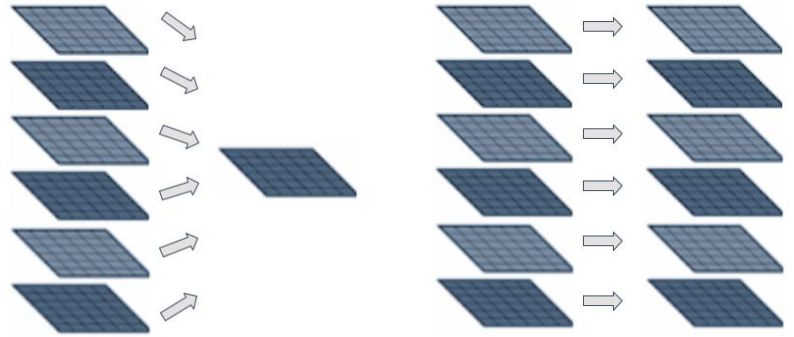
Service definition

```
  inputs:
    aoi_wkt:
      doc: Area of interest
      label: Area of interest
      type: string
    input_files:
      doc: Sentinel-1 SLC acquisition (same track)
      label: Sentinel-1 acquisitions
      type: Directory[]
```

Service parameters

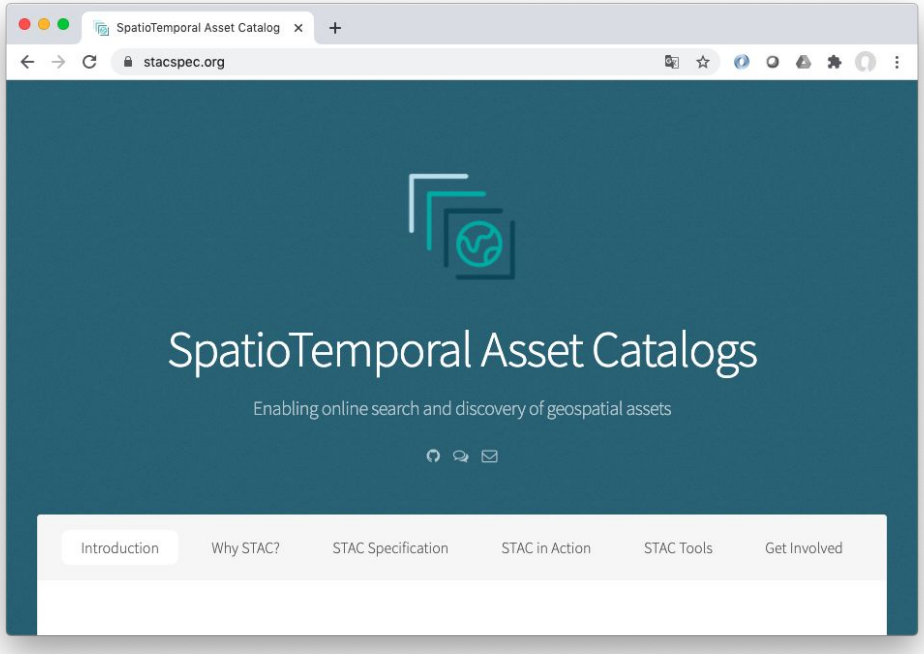
Data Flow Management

- The Platform stages-in the enclosures associated to the input references generating:
  - a local STAC catalog for the fan-in pattern
  - several local STAC catalogs with a single item for the fan-out pattern



- The STAC catalog contains the items (product metadata) and assets (files with manifest and data (.xml, .jp2, tif, etc.))
- The assets have an href relative path which resolves to the full path on the docker mounted volume by cwltool

Data Flow Management



- STAC identifies any file that represents information captured in a certain space and time (spatiotemporal assets)
- A STAC Item is a GeoJSON Feature with additional fields (e.g. time geo), links to related entities and assets (including thumbnails).
- An asset is an object that contains a link to data associated with the Item that can be downloaded or streamed

Data Flow Management

Sentinel-2 fan-in

```
In [50]: 1 s2_search.to_stac(**p)

 * <Catalog id=catalog>
     * <Collection id=collection>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SWC_20181229T112231>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SVB_20181229T112231>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SVC_20181229T112231>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SWB_20181229T112231>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SWB_20181226T113737>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SWC_20181226T113737>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SVB_20181226T113737>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SVC_20181226T113737>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SVB_20181224T111748>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SVC_20181224T111748>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SWC_20181224T111748>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SWB_20181224T111748>
Out[50]: 'stac-catalog/catalog.json'
```

Sentinel-1 interferometric pair

```
In [3]: cat.describe()

 * <Catalog id=catalog>
     * <Collection id=collection>
       * <EOItem id=S1B_IW_SLC__1SDV_20200203T050405_20200203T050432_020100_0260B2_59CC>
       * <EOItem id=S1B_IW_SLC__1SDV_20200122T050405_20200122T050432_019925_025B0B_7EEB>
```

Sentinel-2 fan-out

```
In [46]: 1 s2_search.to_stac(**p)

 * <Catalog id=catalog_0>
     * <Collection id=collection_0>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SWC_20181229T112231>
 * <Catalog id=catalog_1>
     * <Collection id=collection_1>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SVB_20181229T112231>
 * <Catalog id=catalog_2>
     * <Collection id=collection_2>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SVC_20181229T112231>
 * <Catalog id=catalog_3>
     * <Collection id=collection_3>
       * <EOItem id=S2A_MSIL2A_20181229T095411_N0211_R079_T33SWB_20181229T112231>
 * <Catalog id=catalog_4>
     * <Collection id=collection_4>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SWB_20181226T113737>
 * <Catalog id=catalog_5>
     * <Collection id=collection_5>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SWC_20181226T113737>
 * <Catalog id=catalog_6>
     * <Collection id=collection_6>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SVB_20181226T113737>
 * <Catalog id=catalog_7>
     * <Collection id=collection_7>
       * <EOItem id=S2A_MSIL2A_20181226T094411_N0211_R036_T33SVC_20181226T113737>
 * <Catalog id=catalog_8>
     * <Collection id=collection_8>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SVB_20181224T111748>
 * <Catalog id=catalog_9>
     * <Collection id=collection_9>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SVC_20181224T111748>
 * <Catalog id=catalog_10>
     * <Collection id=collection_10>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SWC_20181224T111748>
 * <Catalog id=catalog_11>
     * <Collection id=collection_11>
       * <EOItem id=S2B_MSIL2A_20181224T095419_N0211_R079_T33SWB_20181224T111748>
Out[46]: ['stac-catalog_0/catalog.json',
          'stac-catalog_1/catalog.json',
          'stac-catalog_2/catalog.json',
```

Application Viewpoint

An **Application** needs to:

- Be executed as a command line application.
- Read a STAC local catalog as input manifest
- Create a STAC local catalog as output manifest
- Be executed in a docker container docker image

Application Package Viewpoint

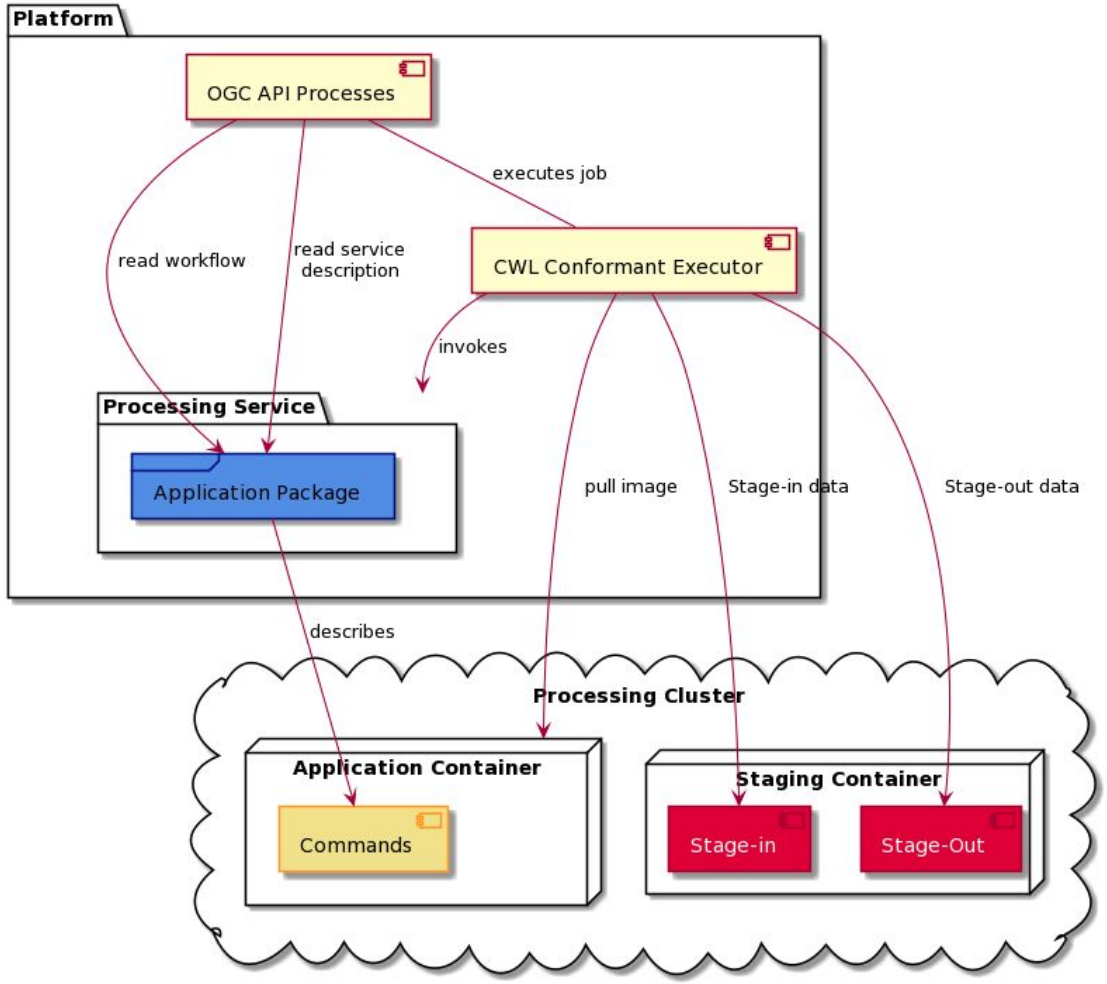
An **Application Package** needs to:

- Be a valid CWL document with a single CWL Workflow Class and at least one Command Line Class
- Define the command line and respective parameters and container for each Command Line
- Define the Application parameters
- Define the requirements for runtime environment

Platform Viewpoint

A **Platform** needs to provide mechanisms to

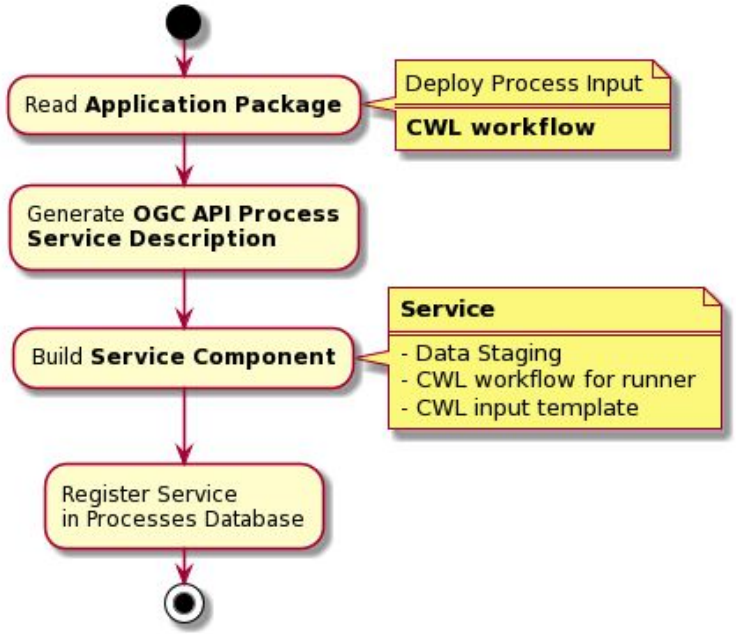
- **deploy** the Application Package
- **execute** the process defined by the Application Package (i.e. creates a new job) with specific parameters



Platform Viewpoint - deployment

For the **deployment**, the platform needs to:

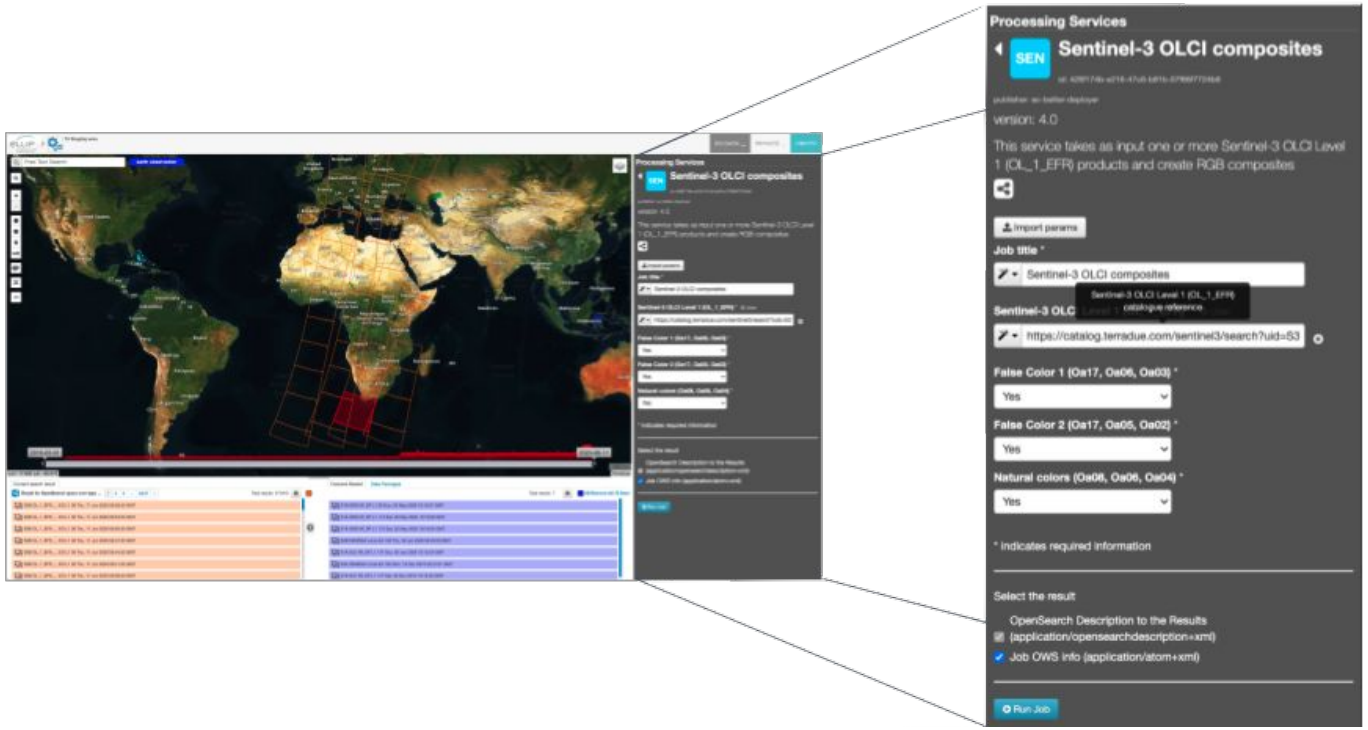
- Accept a Post request with an Application Package (OGC API - Processes)
- Translate Application Package metadata to add a process to a deployed OGC API - Processes server instance
- Translate Application Package Workflow Inputs defined in the CWL document as OGC API - Processes parameters



Platform Viewpoint - execution

For the **execution**, the platform needs to:

- Translate OGC API - Processes execute parameters to the Workflow Inputs defined in the Application Package (CWL document)



Platform Viewpoint - execution

For the **execution**, the platform needs to:

- Translate OGC API - Processes execute parameters to the Workflow Inputs defined in the Application Package (CWL document)
- If applicable, execute the data stage-in for the input EO products
- Orchestrate and execute CWL
- Translate output to OGC API process outputs