TERRADUE

Advancing Earth Science

www.terradue.com

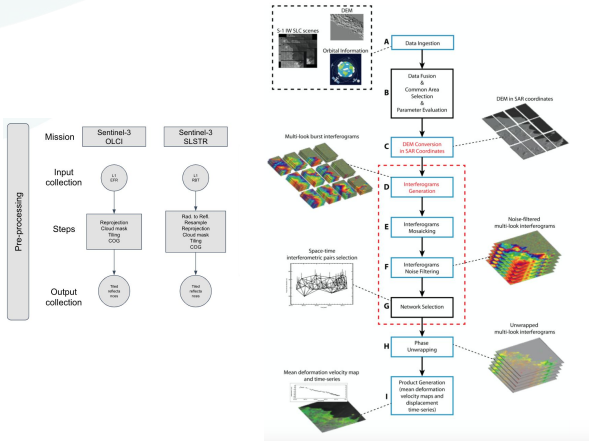
Ellip Studio

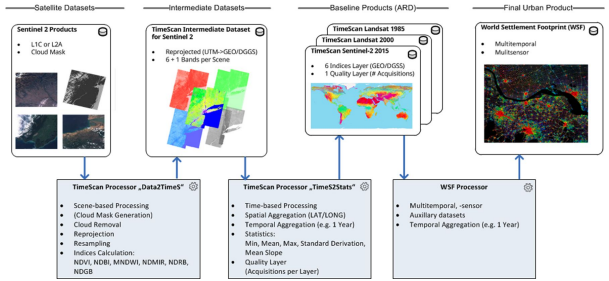
A JupyterLab Environment for developing Cloud-ready Earth Observation Applications

May 2022

TerraDue
Advancing Earth Science

EO Application Package





- An Earth Observation Application is set of command-line tools with numeric, textual and EO data parameters organized as a computational workflow
- An Application Package uses an explicit language that describes the input and output interface of the computational workflow and the orchestration of its command-line tools.
- The Application Package guarantees the automation, scalability, reusability, portability of the Application while also being workflow-engine and vendor neutral.

New OGC Best Practice for EO Application Packages

Open Geospatial Consortium

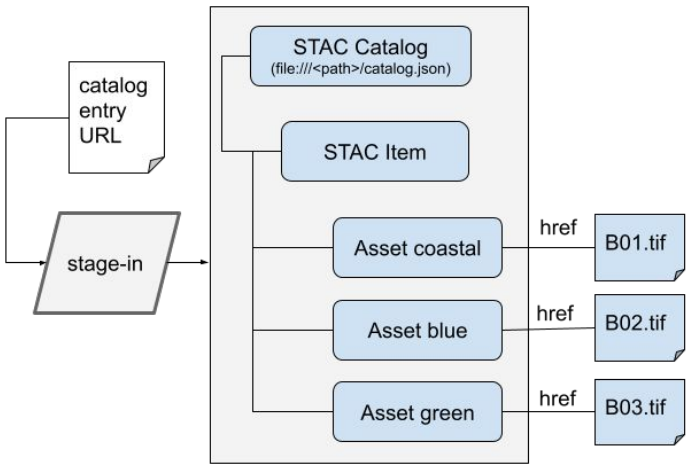
EO Application Package

- The command-line tools (e.g. Python, shell script, C++) and their dependencies are containerized and registered in a container registry
- The computational workflow input and output interfaces and the orchestration of its command-line tools are described with Common Workflow Language (CWL)

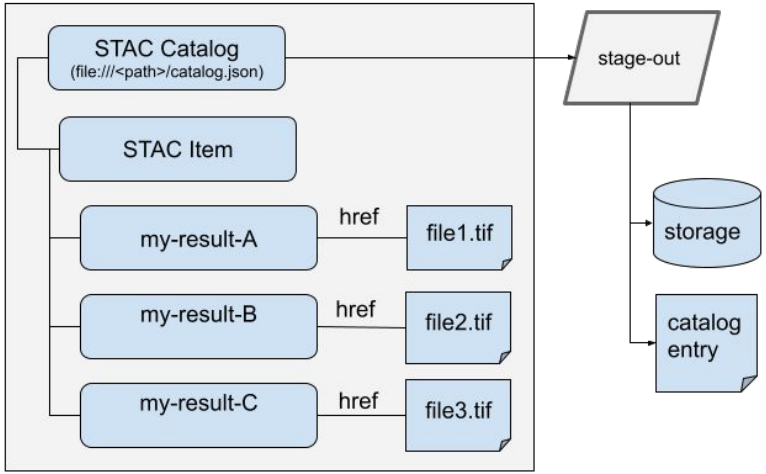
The Common Workflow Language (CWL) is an ***open standard*** for describing analysis workflows and tools in a way that makes them ***portable*** and ***scalable*** across a variety of ***software and hardware environments***, from workstations to cluster, cloud, and high performance computing environments.

EO Application Package

- The computational workflow data interfaces use the Spatio Temporal Asset Catalog (STAC) to describe the EO data inputs and generated results.

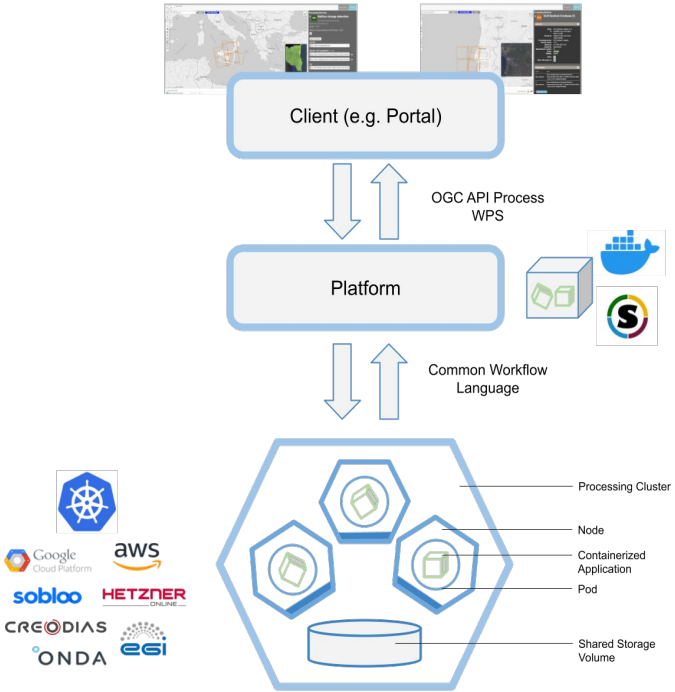


Stage-in



Stage-out

EO Application Package



- The Platform takes the CWL application package and exposes an OGC API Processes processing service.
- The Platform provides the automation, scalability, reusability, portability by converting the OGC API Processes execution request into a CWL execution using a runner and the computing resources of the selected provider.

TerraDue
Advancing Earth Science

An example with gdal

```
$graph:
- class: CommandLineTool
  id: crop-cl
  requirements:
    DockerRequirement:
      dockerPull: docker.io/osgeo/gdal
    InlineJavascriptRequirement: {}
  baseCommand: gdal_translate
  arguments:
  - -projwin
  - $( inputs.bbox.split(",")[0] )
  - $( inputs.bbox.split(",")[3] )
  - $( inputs.bbox.split(",")[2] )
  - $( inputs.bbox.split(",")[1] )
  - -projwin_srs
  - "EPSG:4326"
  - $( "/vsicurl/" + inputs.cog )
  - cropped.tif
  inputs:
    cog:
      type: string
    bbox:
      type: string
  outputs:
    cropped_tif:
      outputBinding:
        glob: '*.tif'
      type: File
cwlVersion: v1.0
```

```
INFO [job crop-cl] /tmp/6e0o3onn$ docker \
    run \
    -i \
    --mount=type=bind,source=/tmp/6e0o3onn,target=/MtUYrD \
    --mount=type=bind,source=/tmp/fk4pojtj,target=/tmp \
    --workdir=/MtUYrD \
    --read-only=true \
    --user=1000:1000 \
    --rm \
    --cidfile=/tmp/tn7sa93x/20220525113057-272748.cid \
    --env=TMPDIR=/tmp \
    --env=HOME=/MtUYrD \
    docker.io/osgeo/gdal \
    gdal_translate \
    -projwin \
    136.983 \
    -35.831 \
    137.112 \
    -35.92 \
    -projwin_srs \
    EPSG:4326 \

/vsicurl/https://sentinel-cogs.s3.us-west-2.amazonaws.com/sentinel-s2-l2
a-cogs/53/H/PA/2021/7/S2B_53HPA_20210703_0_L2A/B02.tif \
    cropped.tif
```

Skills and tooling

Skills

- YAML
- Containers (docker files, docker build, tags, etc.)

Tooling

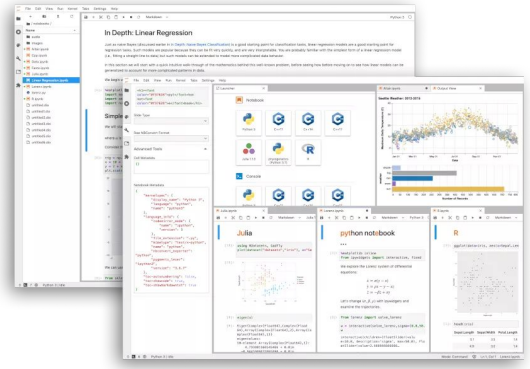
- A container engine: docker or podman
- A CWL runner: cwltool, calrissian (k8s)
- An IDE: VS Code or Theia/Coder (in the Cloud)
- An object storage (S3)
- Access to a container registry (e.g. docker.io, quay.io Gitlab, Github)
- Access to Continuous Integration service (e.g. Gitlab CI, Github Actions, Jenkins, etc.)
- Access to a Package Registry (e.g. Gitlab, Github, Artifactory)

How can we provide these tools in a fully fledged IDE in the Cloud ?

TERRAQUE
Advancing Earth Science

Jupyter Notebooks ubiquitous

Providing JupyterLab as part of a service offering is trivial nowadays



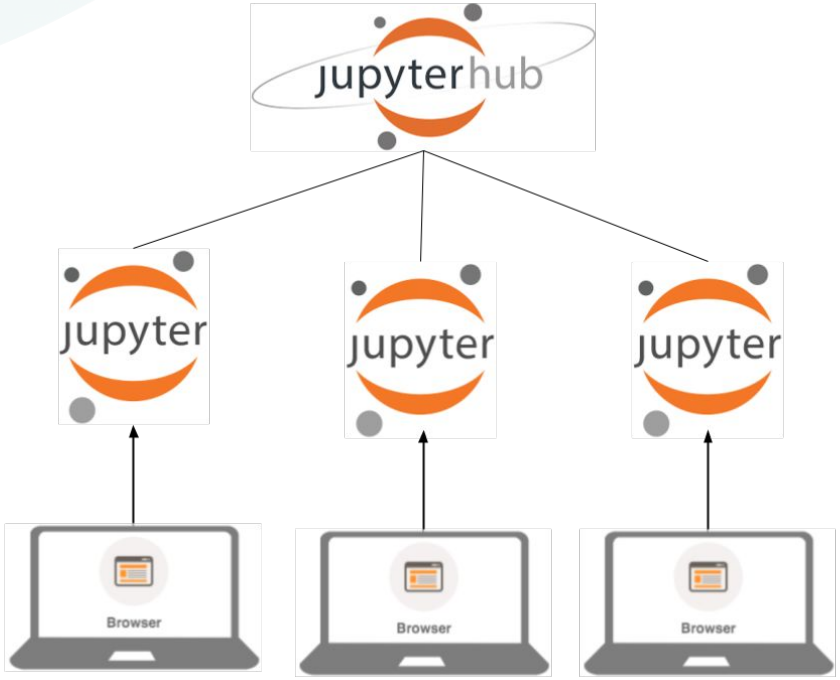
Typical setup:

- JupyterHub spawns user dedicated JupyterLab instances
- Workspace persistence of a few tens of gigas
- Pre-installed tools in base image

But beyond the Notebook experience, it's somehow a poor IDE

TerraDue
Advancing Earth Science

JupyterHub for isolated servers



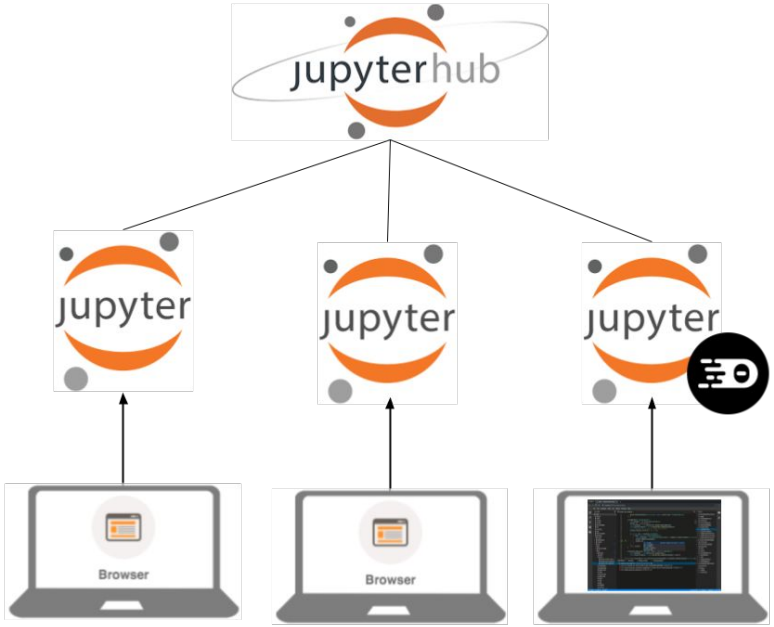
At scale we often rely on JupyterHub to provision users with isolated JupyterLab instances.

Coupled with kubernetes and kubespawner we get isolated and dedicated JupyterLab instances

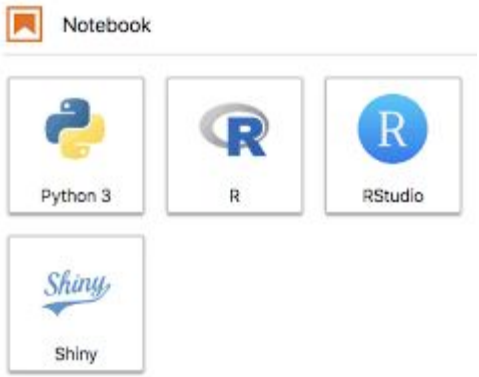
This provides JupyterLab but lacks a fully fledged IDE

TerraDue
Advancing Earth Science

SaaS with JupyterLab



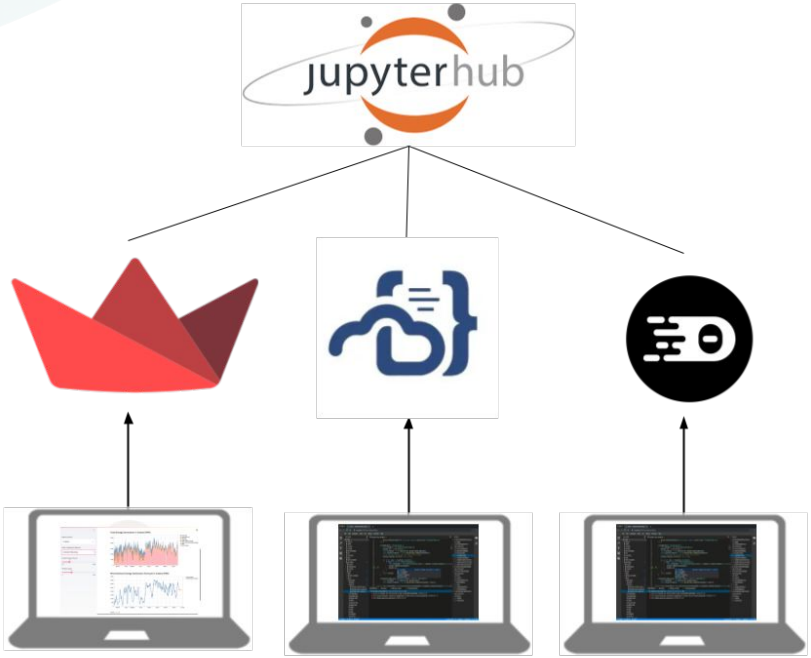
With extensions, the JupyterLab instance can proxy other applications like RStudio, Shiny Server, Theia IDE or Code Server IDE



This provides JupyterLab and fully fledged IDE

TERRAQUE
Advancing Earth Science

SaaS with JupyterHub



With other extensions, the **JupyterHub** instance can launch other applications than JupyterLab using dedicated containers:
IDEs with Code Server or Theia, dashboards with e.g. Streamlit

This provides SaaS for isolated applications

Extend for EO app development

Storage

Access to object storage (s3) for EO reference and test dataset

EO toolboxes

No need to pre-install the EO toolboxes in base image

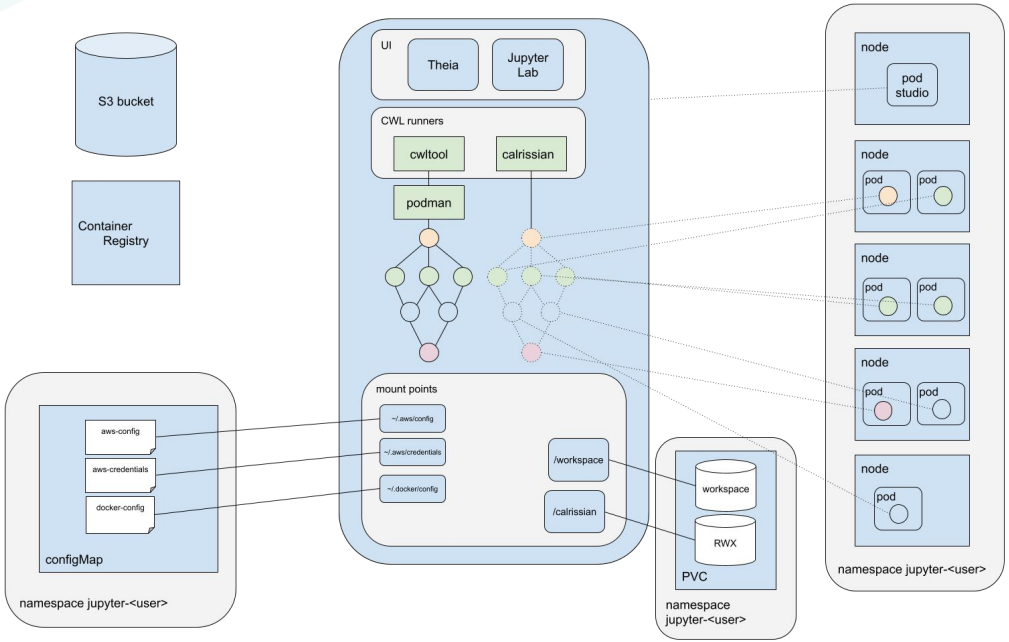
Everything done with conda/mamba

Containers

Run containers with CLI and additional tooling

Provide additional control over advanced tools management and storage

Ellip Studio



Fully integrated with k8s:

- user data via configmaps
- user workspace persisted with PVC
- RWX volume for CWL horizontal scaling

Provide vertical and horizontal scalability with kubernetes to develop and test EO application packages

TERRADUE
Advancing Earth Science

ELLIP
Collaborative Earth Science Services

Ellip Studio - conclusion

- SaaS with JupyterLab and Code Server
- Advanced tooling:
  - Container engine with podman
  - EO toolboxes (autonomously installed)
  - Object Storage tools
  - CWL runners: cwltool and calrissian
- Storage
  - Persistent workspace
  - Object storage
  - RWX storage for CWL horizontal scaling
- Container registry, Continuous Integration

Looking forward hearing from you!

https://www.terradue.com

Fabrice Brito

fabrice.brito@terradue.com