The Journal of Open Source Software

JupyterGIS: A Collaborative GIS Environment for JupyterLab

Martin Renou [1], **Arjun Verma** [1], **Matt Fisher** [2], **Gregory Mooney** [1], **Matthias Meschede** [1], **Nicolas Brichet** [1], **David Brochart** [1], **Nakul Verma** [4], **Anne Fouilloux** [3], **Sylvain Corlay** [1¶], **and Fernando Pérez** [2]

1 QuantStack, France **2** Eric and Wendy Schmidt Center for Data Science & Environment, United States **3** LifeWatch ERIC, Spain **4** Indira Gandhi National Open University ¶ Corresponding author

DOI: 10.21105/joss.10383

Software

- Review
- Repository
- Archive

Editor: Abhishek Tiwari

Reviewers:

- @robertmartin8
- @ianhi

Submitted: 28 December 2025
Published: 19 August 2026

Summary

JupyterGIS is a JupyterLab (Kluyver et al., 2018) extension that enables web-based Geographic Information System (GIS) workflows. It provides a familiar GIS interface inspired by traditional desktop GIS tools, real-time collaborative editing, and a Python API for programmatic control, making it a powerful tool for geospatial data analysis and visualization. JupyterGIS supports a wide range of geospatial data formats, including GeoTIFFs and Cloud-Optimized GeoTIFFs, Shapefile, GeoParquet, and PMTiles, and provides advanced features such as symbology editing, spatio-temporal animations, and a browser-based processing toolbox powered by WebAssembly (WASM) builds of GDAL (Rouault et al., 2026).

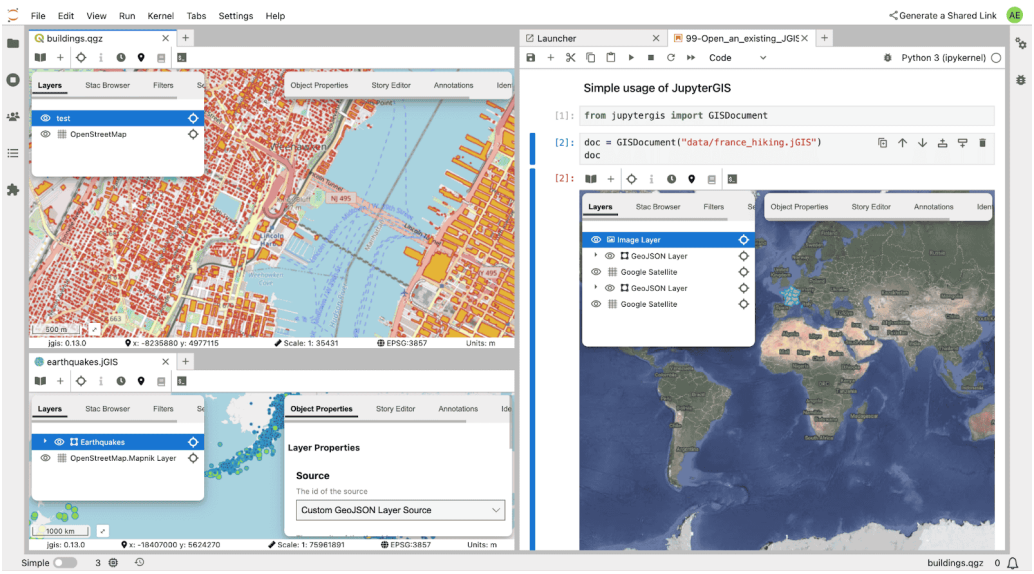

Figure 1: Screenshot of JupyterGIS in action

Statement of Need

Geospatial data analysis and visualization are essential in fields such as environmental science, urban planning, and disaster management. However, traditional GIS tools often lack **real-time collaboration** and integration with computational **notebooks**. JupyterGIS addresses these gaps by:

- Enabling **real-time collaborative editing** for GIS projects.
- Providing **interactive maps and geospatial visualizations within Jupyter notebooks**.
- Supporting **programmatic control** via a Python API, allowing for automation and reproducibility.
- Offering **browser-based access to GIS workflows**, reducing the need for desktop software.

State of the Field

The geospatial software ecosystem comprises a broad diversity of tools addressing specific needs, including proprietary solutions and open-source alternatives, cloud-based platforms, and notebook-integrated workflows.

However, none of the existing open-source offerings addresses the growing demand for real-time collaboration.

Closed-Source Desktop Solutions

Esri's ArcGIS remains the dominant proprietary GIS platform, offering comprehensive tools for data management, analysis, and visualization. While ArcGIS Online provides cloud-based collaboration features, it is not as instantaneous. Edits are synchronized at scheduled intervals or manually, not in real time.

Open-Source Desktop Solutions

QGIS is the leading open-source desktop GIS, renowned for its extensibility, support for diverse data formats, and vibrant community.

It provides a powerful alternative to proprietary software but does not allow real-time collaborative editing of GIS documents. As a desktop application, it must be installed on the user's device.

Closed-Source Cloud-Based Platforms

Google Earth Engine enables large-scale geospatial analysis in the cloud. However, its focus on script-based workflows and lack of interactive, collaborative editing make it less suitable for teams needing real-time collaboration.

Felt is a collaborative web-based mapping tool focused on ease of use for non-technical users. It includes a REST API and a Python client to add layers to maps.

In-Notebook Tools

Libraries like ipyleaflet and folium bring interactive mapping to Jupyter notebooks. While useful for exploratory analysis, these tools lack a graphical user interface for non-developers wishing to create GIS documents with advanced layer styling.

JupyterGIS

JupyterGIS brings a collaborative, desktop-style GIS experience to the web, enabling real-time coediting of GIS documents directly in JupyterLab.

Supported Layer Types

JupyterGIS supports a broad range of **geospatial data formats**, including vector formats (**GeoJSON**, **Shapefile**, **GeoParquet**), raster formats (Cloud-Optimized GeoTIFFs), as well as **raster and vector tile layers**.

Real-time Collaboration and Annotation

JupyterGIS enables live multi-user collaboration: track cursors, follow others' views, and annotate maps in real time.

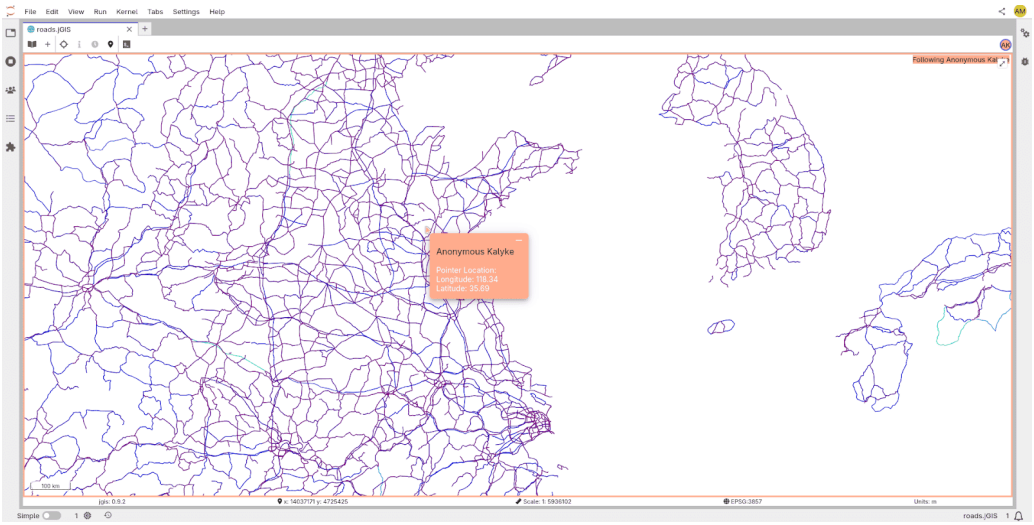

Figure 2: Screenshot of the follow mode, tracking a user cursor and viewport

Python API

JupyterGIS provides a **Python API** for editing GIS documents from Jupyter notebooks and consoles, integrated into Jupyter's display system. The Python API operates on the shared document like a collaborator.

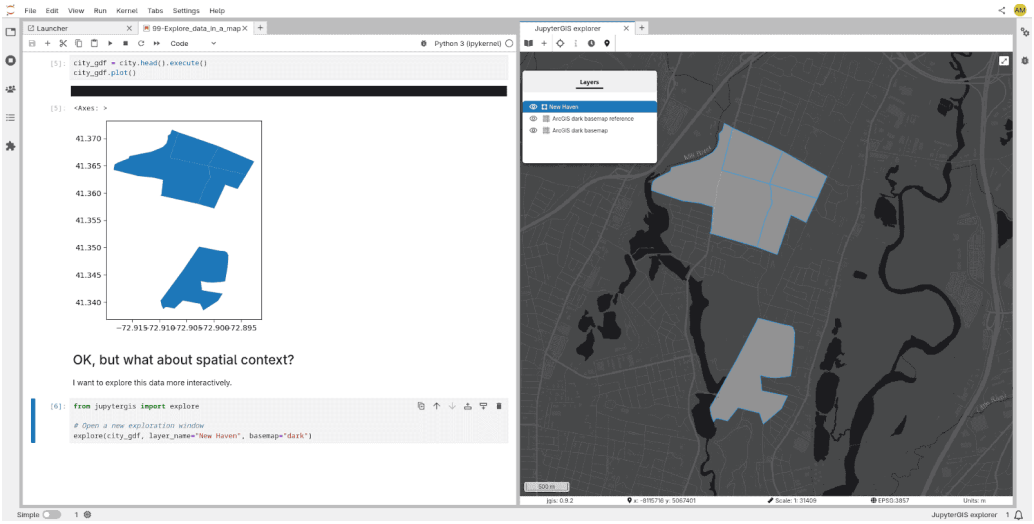

Figure 3: Screenshot of the Python API

Deployment Options

JupyterGIS can be deployed with any Jupyter-based environment, from **JupyterHub** to **JupyterLite**.

Extensibility

JupyterGIS supports extensions, including the **JupyterGIS-tiler** plugin, which enables Xarray variables to be displayed as map layers—bridging JupyterGIS with the Pangeo ecosystem.

The Processing Toolbox

JupyterGIS includes a set of commonly used processing operations such as buffering, dissolving, centroid computation, and convex hulls.

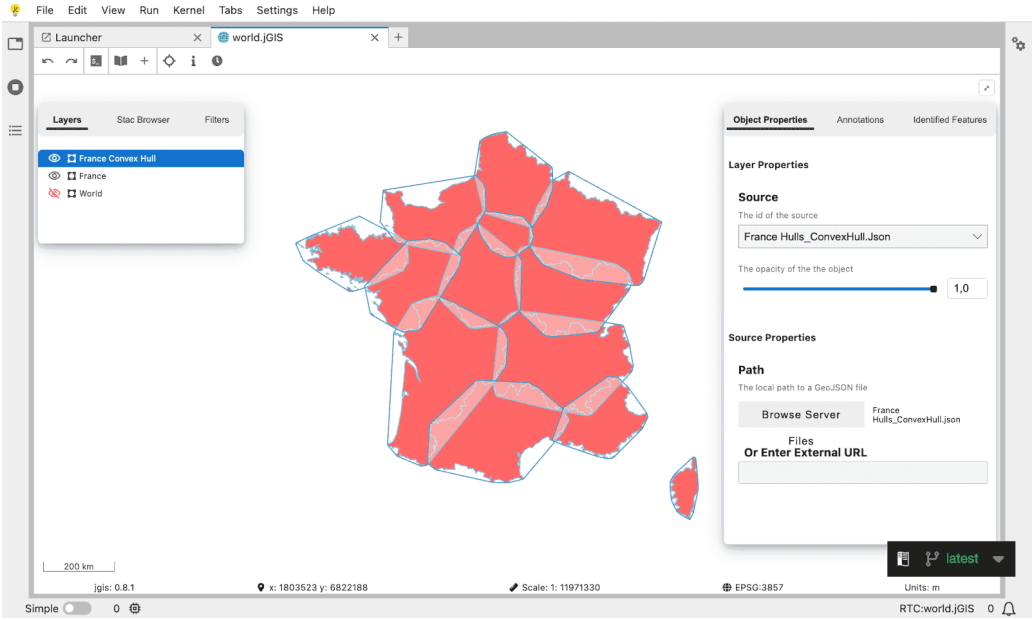

Figure 4: Screenshot showcasing the processing toolbox

Support for QGIS Project Files

JupyterGIS allows editing QGIS project files (`.qgz` and `.qgs`) directly, with a partial support of the QGIS features.

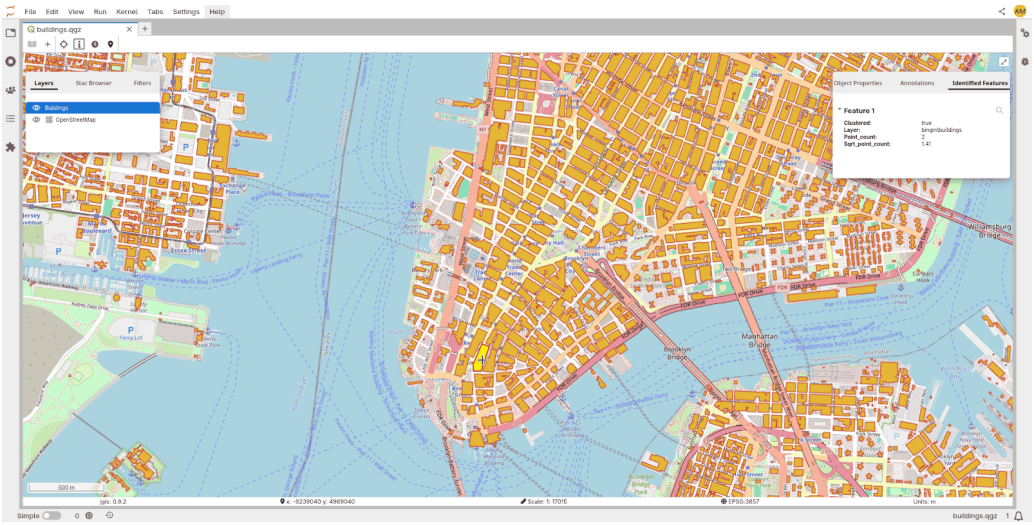

Figure 5: Screenshot of JupyterGIS operating on a QGIS file

Advanced Symbology

The symbology panel enables users to style vector layers with graduated, categorized, canonical, and heatmap renderers, while raster layers support single-band and multi-band visualization.

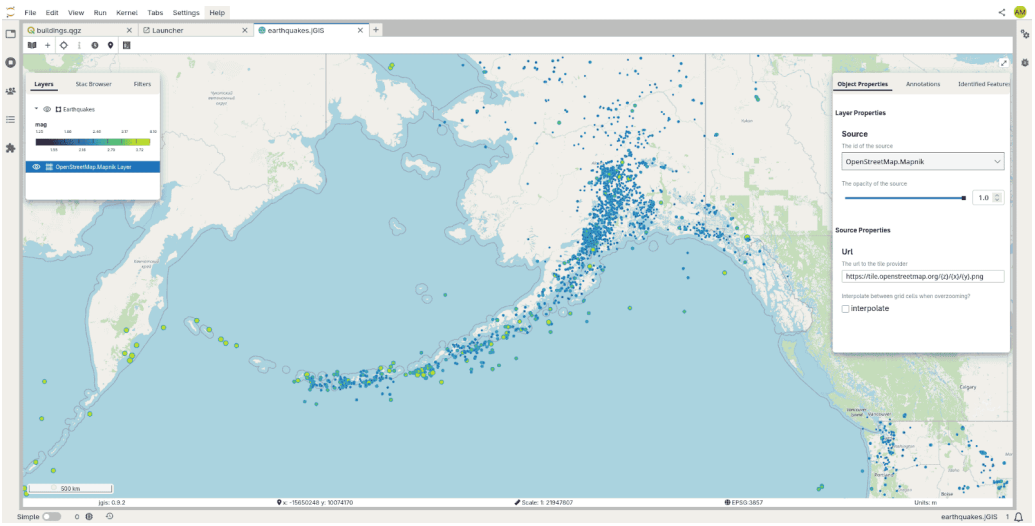

Figure 6: Screenshot of JupyterGIS advanced symbology: a graduated colormap of earthquake magnitudes

Interactive Tools and Dynamic Visualizations

- The "identify" tool allows clicking on features to view their attributes.
- Time-dependent datasets can be animated, supporting visualization of changes across time.

Story Maps

JupyterGIS's Story Map feature lets users combine interactive maps with text, images, and multimedia to create engaging, narrative-driven visualizations.

Software Design

Collaborative Editing as a First-Class Requirement: Shaping the JupyterGIS Document Model

Collaborative editing has transformed how teams work, replacing slow email exchanges with real-time collaboration. Looking ahead, the potential of coediting extends beyond text documents, especially in designing complex systems that require diverse expertise, such as climate modeling, agriculture, and urban planning.

Users will soon expect co-editing as the norm:

- Self-collaboration: Effortless editing across devices and windows, without conflicts.
- Real-time sharing: collaboration via a link, with zero risk of data corruption.

These capabilities should be core to any application, not bolted on later.

Building upon the JupyterLab Application Framework

Professionals depend on advanced authoring tools — IDEs, CAD modelers, and GIS applications — for daily productivity. They have high expectations for tools that they use for extended

periods of time: extensibility, customization, theming, internationalization, and scriptability are strong requirements.

JupyterLab delivers these features as a robust framework, backed by a large user base and extension ecosystem. It supports collaborative editing via CRDTs (Conflict-free Replicated Data Types) (Shapiro et al., 2011) through the Yjs framework (Nicolaescu et al., 2015). JupyterGIS builds on this foundation, integrating seamlessly with the Jupyter ecosystem and enabling serverless deployments via JupyterLite.

The JupyterGIS Stack

JupyterGIS leverages the following technologies:

Frontend	JupyterLab application framework
Collaboration	Collaborative editing with Yjs and PyCRDT (Nicolaescu et al., 2015)
Processing	GDAL-based (Rouault et al., 2026) toolbox
Visualization	OpenLayers (OpenLayers Contributors, 2025) for map rendering

Research Impact

Deployments on Research Infrastructure

JupyterGIS has been deployed across several major **research infrastructures**:

- It is available on the Copernicus Data Space Ecosystem (CDSE).
- It has been integrated with the Open OnDemand portal (Hudak et al., 2018) and deployed at the University of Oslo.
- JupyterGIS is integrated into the Galaxy Toolbox (Abueg et al., 2024) and deployed on the Galaxy Europe deployment of EOSC at usegalaxy.eu. It will soon be installed on the Earth Science thematic node of EOSC. More details on the Galaxy integration are available in an article published on the Galaxy project's blog.

Public Deployments

JupyterGIS is accessible through multiple public deployments:

- Ready-to-use JupyterGIS environments are available via Binder (Jupyter et al., 2018), with direct access provided in the JupyterGIS GitHub repository.
- The repository also features a **JupyterLite-based deployment**, a fully static solution that runs in the browser using **WebAssembly** for Python execution. This approach eliminates the need for cloud infrastructure, enabling **scalable and lightweight deployments**.

Supporting Scientific Publications

JupyterGIS powers an interactive map of global CO2 storage potential in sedimentary basins, hosted by IIASA and deployed via JupyterLite as supplementary material for a Nature article (Gidden et al., 2025).

Community and Contributions

Contributing to JupyterGIS

JupyterGIS is available under the BSD 3-Clause License. Contributions are welcome on the GitHub repository. The documentation is available online on Read The Docs. We host community discussions on a public channel.

The GeoJupyter Initiative

JupyterGIS has been incorporated as the first and central component of a broader initiative: GeoJupyter, a community-driven effort to reimagine geospatial computing for education, research, and industry.

Future Work

Our roadmap includes the following future developments:

Integration with openEO

JupyterGIS will soon support openEO (Schramm et al., 2021) process graphs as a supported layer type, dynamically rendered as XYZ tiles in JupyterGIS. This requires extending both the document model and the Python API to embed openEO process graphs directly within JupyterGIS documents.

This integration seamlessly connects cloud-based geospatial processing with collaborative document editing.

An R API

To extend JupyterGIS to the R ecosystem, we will develop an R API mirroring the functionality of the Python API.

This API will interact with the collaborative editing framework and the underlying data model, just as the Python API does. The primary requirement is to create R bindings for the y-crdt Rust library, which underlies the collaborative data model in the backend and the Python bindings.

JupyterLite-AI

We will integrate JupyterGIS with JupyterLite-AI by exposing JupyterGIS features as tools to Large Language Models (LLMs). By leveraging the JupyterLab application framework, we will ensure that all user actions are backed by JupyterLab commands, which are exposed to JupyterLite-AI's tool calling system.

Acknowledgments

JupyterGIS was initially developed through a collaboration between **QuantStack**, the **Simula Research Laboratory**, and the **Eric and Wendy Schmidt Center for Data Science & Environment at UC Berkeley** (DSE), with additional contributions from community members.

- **QuantStack** and **Simula Research Laboratory** received funding from the European Space Agency (**ESA**) through the Open Call for EO Innovation.
- QuantStack secured additional funding from the Centre National d'Études Spatiales (**CNES**) to develop the STAC browser and story maps feature.
- QuantStack contributed further to the project through unfunded efforts.
- The **DSE** funded the contributions of its researchers to the project.

AI Usage Disclosure

The development of JupyterGIS relied entirely on **human expertise** and traditional software engineering practices. While we leveraged developer productivity tools, such as IDE features

for code auto-completion and suggestions, every contribution, including code, documentation, and design, underwent review by the core team.

This article was written entirely by humans, though we used productivity tools for grammatical corrections and proofreading.

References

Abueg, L. A. L., Afgan, E., Allart, O., Awan, A. H., Bacon, W. A., Baker, D., Bassetti, M., Batut, B., Bernt, M., Blankenberg, D., Bombarely, A., Bretaudeau, A., Bromhead, C. J., Burke, M. L., Capon, P. K., Čech, M., Chavero-Díez, M., Chilton, J. M., Collins, T. J., … Zoabi, R. (2024). The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. *Nucleic Acids Research*, *52*(W1), W83–W94. https://doi.org/10.1093/nar/gkae410

Gidden, M. J., Joshi, S., Armitage, J. J., Christ, A.-B., Boettcher, M., Brutschin, E., Köberle, A. C., Riahi, K., Schellnhuber, H. J., Schleussner, C.-F., & Rogelj, J. (2025). A prudent planetary limit for geologic carbon storage. *Nature*, *645*(8079), 124–132. https://doi.org/10.1038/s41586-025-09423-y

Hudak, D., Johnson, D., Chalker, A., Nicklas, J., Franz, E., Dockendorf, T., & McMichael, B. L. (2018). Open OnDemand: A web-based client portal for HPC centers. *Journal of Open Source Software*, *3*(25), 622. https://doi.org/10.21105/joss.00622

Jupyter, P., Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., Holdgraf, C., Kelley, K., Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B., & Willing, C. (2018). Binder 2.0 - reproducible, interactive, sharable environments for science at scale. *Proceedings of the 17th Python in Science Conference*, 113–120. https://doi.org/10.25080/majora-4af1f417-011

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & Jupyter development team. (2018). Jupyter notebooks — a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. https://doi.org/10.3233/978-1-61499-649-1-87

Nicolaescu, P., Jahns, K., Derntl, M., & Klamma, R. (2015). Yjs: A framework for near real-time P2P shared editing on arbitrary data types. *Engineering the Web in the Big Data Era*. https://doi.org/10.1007/978-3-319-19890-3_55

OpenLayers Contributors. (2025). *OpenLayers*. https://openlayers.org

Rouault, E., Warmerdam, F., Schwehr, K., Kiselev, A., Butler, H., Łoskot, M., Szekeres, T., Tourigny, E., Landa, M., Miara, I., Elliston, B., Chaitanya, K., Plesea, L., Morissette, D., Jolma, A., Dawson, N., Baston, D., Stigter, C. de, & Miura, H. (2026). *GDAL* (Version v3.13.2). Zenodo. https://doi.org/10.5281/zenodo.20700933

Schramm, M., Pebesma, E., Milenković, M., Foresta, L., Dries, J., Jacob, A., Wagner, W., Mohr, M., Neteler, M., Kadunc, M., Miksa, T., Kempeneers, P., Verbesselt, J., Gößwein, B., Navacchi, C., Lippens, S., & Reiche, J. (2021). The openEO API–harmonising the use of Earth Observation cloud services using virtual data cube functionalities. *Remote Sensing*, *13*(6), 1125. https://doi.org/10.3390/rs13061125

Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict-free replicated data types. In X. Défago, F. Petit, & V. Villain (Eds.), *Stabilization, safety, and security of distributed systems* (pp. 386–400). Springer Berlin Heidelberg.