Towards Standardization of the Earth Observation Data Product Supply Chain – Are OCI Artifacts the Key to Ubiquitous and Scalable EO Data Handling?

Stefan Achtsnit[1,2], Martin Rösel[1], Stephan Meißl[2]

[1]Versioneer GmbH, Lower Austria, Austria – (stefan.achtsnit, martin.roesel)@versioneer.at
[2]EOX IT Services GmbH, Vienna, Austria – (stefan.achtsnit, stephan.meissl)@eox.at

Keywords: Open Container Initiative (OCI), OCI Registries and Artifacts, Layered Data Storage, Data Supply Chain

Abstract

The exponential growth of Earth Observation (EO) data generated by satellites demands scalable, efficient, and interoperable methods not only for storing and managing, but also for distributing data packages tailored to diverse use cases. The Open Container Initiative (OCI) registries, together with the OCI artifact specifications, present a promising framework for packaging, exchanging, and managing EO-derived and combined datasets. Originally developed for software containers, OCI registries offer key capabilities such as content-addressable storage, data integrity verification, cryptographic attestation, layered packaging, and version control. A notable advantage is their ability to act as access gateways—enforcing access control at the artifact level without requiring direct exposure of the underlying storage backends (e.g., S3, GCS, Azure Blob, NFS, Ceph, IPFS). The ubiquity of OCI registries—spanning public platforms, managed enterprise services, and open-source deployments—makes them a practical foundation for distributing EO data across heterogeneous environments without custom infrastructure. This paper investigates the applicability of today's OCI registry ecosystem to EO data pipelines, evaluating both strengths and current limitations in handling large, complex, and dynamic datasets. We explore design conventions and layout strategies to align EO products with the OCI artifact model, with a focus on metadata representation, access efficiency, and storage reuse. By comparing self-contained data packages with modular, layered asset stores, we highlight trade-offs in retrieval performance, interoperability, and client complexity. Recent trends in machine learning model distribution further underscore the growing relevance of OCI-based artifacts for scientific and geospatial workflows. Ultimately, this research positions OCI artifacts as a viable foundation for scalable, standards-aligned, and interoperable EO data handling—paving the way toward more streamlined and resilient data supply chains in the EO domain.

1. Introduction

The increasing volume and complexity of Earth Observation (EO) data—driven by advancements in satellite constellations, sensor technologies, and remote sensing methodologies—pose significant challenges for efficient distribution, management, and utilization. EO data products, typically comprising high-resolution, time-series, and spatially rich information, must be delivered in a way that meets the diverse and often stringent requirements of different use cases. In many instances, raw EO data is reshaped, transformed, or fused with complementary datasets—such as meteorological records, environmental indices, and in situ measurements—to derive actionable insights tailored to specific analytical tasks.

Effective distribution in this context demands not only scalability, low overhead, and rapid access but also robust support for data provenance, lineage, and compliance as part of the data product supply chain. These aspects are critical for ensuring reproducibility of analyses, validating the integrity of derived products, and meeting regulatory and institutional mandates—especially as EO data increasingly underpins decision-making in domains such as climate monitoring (WMO, 2024), disaster response (Rolla, A., 2025), and precision agriculture (FAO, n.d.). Traditional data distribution methods, however, often depend on bespoke architectures and ad hoc tooling, which can hinder interoperability, complicate maintenance, and inhibit broader adoption across heterogeneous environments.

The Open Container Initiative (OCI) presents a promising alternative. Originally developed to package and distribute software applications, OCI registries provide a standardized and scalable framework with built-in features such as versioning, immutability, layered storage, and cryptographic attestation. These capabilities enable secure, auditable, and traceable distribution—qualities that are just as valuable in data-intensive domains like Earth Observation (EO) as they are in the software supply chain.

Although the use of OCI registries for EO datasets is still emerging, the ubiquity and maturity of OCI infrastructure make it an appealing foundation for data packaging and distribution. This existing infrastructure landscape spans:

- public cloud platforms like Docker Hub, AWS ECR,
- managed enterprise services e.g., Quay.io (Quay, n.d.),
- open-source deployments e.g., Harbor (Harbor, n.d.)

and supports a range of storage backends—including centralized cloud storage, on-premises systems, or even decentralized platforms such as IPFS (IPFS, n.d.). Together, these options support flexible deployment across centralized cloud environments, on-premises systems, and emerging distributed architectures.

We argue that OCI registries are particularly well-suited to EO data workflows, offering a structured and observable mechanism for organizing and managing datasets used in domain-specific analyses, automated inference pipelines, compliance archiving, and AI/ML development. By leveraging existing tooling and standards—already deeply integrated into modern software development and operations ecosystems—EO data producers and consumers can reduce the operational friction and technical

debt associated with maintaining bespoke distribution systems. This paper investigates how EO data can be structured and packaged as OCI artifacts to conform to the OCI artifact specification while preserving efficiency across diverse registry implementations. We focus on practical strategies for organizing datasets to optimize retrieval speed, storage efficiency, and scalability, with particular attention to the varying constraints and behaviors of different OCI registries. Rather than modifying OCI registry technology itself, we explore how EO data packaging can be aligned with the capabilities and limitations of existing OCI infrastructure. Based on a benchmark artifact set and experimental evaluation, we provide guidance for practitioners seeking to adopt OCI-based workflows for EO data distribution.

Our findings aim to motivate the EO community with practical strategies for harnessing existing OCI registry solutions to support real-time analysis, secure data access, and sustainable long-term data management—particularly in distributed and decentralized environments. By framing OCI as a foundational framework for EO data handling, we highlight its potential to drive greater standardization, interoperability, and scalability across the Earth Observation ecosystem.

2. Data and Materials

The efficient and scalable distribution of EO data remains a central challenge in modern remote sensing workflows. These pipelines must accommodate large volumes of geospatial and temporal data, often under strict requirements for reproducibility, traceability, and accessibility—principles actively promoted by initiatives such as EarthCODE (Anghelea et al., 2024). Conventional architectures typically rely on cloud-based object storage systems, complemented by metadata catalogs—most notably the SpatioTemporal Asset Catalog (STAC, 2023), which provides a standardized interface for describing and indexing geospatial assets. While widely adopted, these systems often lack native mechanisms for version control, content reuse, and coherent packaging of heterogeneous data. These capabilities—established in software engineering and ML model versioning—are becoming essential for scientific reproducibility and operational consistency.

In recent years, the need for structured and reproducible data access has become increasingly urgent in analytics and machine learning workflows. Relying solely on live, mutable data streams—such as satellite imagery, meteorological feeds, or in situ measurements accessed via HTTP or S3 APIs—is no longer sufficient. Users increasingly demand materialized and curated datasets: well-defined, harmonized collections that are immediately usable for downstream analysis or model training. Platforms like Hugging Face Datasets (HF Datasets, n.d.) have demonstrated the effectiveness of this paradigm by offering versioned datasets with structured metadata and reproducibility guarantees. Domain-specific initiatives like the Earth Observation Training Data Lab (EOTDL, n.d.) are now emerging to bring similar capabilities to the Earth Observation community. To ground our discussion in a practical and representative example, we focus on the Panoptic Agricultural Satellite Time Series (PASTIS, 2024) dataset (Sainte Fare Garnot & Landrieu, 2021)—an arbitrarily selected yet illustrative case. It integrates multiple Earth Observation modalities:

- Optical time-series data from Sentinel-2,
- Radar time-series data from Sentinel-1,
- Very High Resolution imagery from SPOT satellites,
- Curated annotations, including label masks and semantic classifications.

All data is spatially tiled and georeferenced. Sentinel data is stored in NumPy array format, SPOT imagery in TIFF format, and annotations in structured formats suitable for semantic labeling tasks. With a total size of approximately 80 GB across four distinct regions in France, the dataset exemplifies several common challenges in EO workflows: multi-modal integration, heterogeneous formats, tile-based partitioning, and the coexistence of dense temporal sequences with high-resolution spatial snapshots.

All benchmarking, packaging experiments, and structural evaluations in this study are based on two intentionally contrasting partitioning strategies applied to the PASTIS dataset. However, the packaging approaches we explore are data-agnostic and readily applicable to other EO datasets, regardless of their origin or internal format. For example:

- Gridded data could be stored as NetCDF or Zarr chunks instead of NumPy arrays.
- Imagery could be provided in formats such as JPEG2000 or Cloud Optimized GeoTIFFs (COGs) rather than standard TIFF.
- Auxiliary sources—such as meteorological feeds or in-situ observations—can be integrated just as easily.

That said, these format variations are not the focus of this study; they are mentioned to emphasize the broader applicability of the methods presented.

The packaged OCI artifacts produced in this work are immutable, content-addressable, and versioned data units. While OCI was originally designed for containerized software delivery, registries have since evolved to support arbitrary digital artifacts, enabling new applications well beyond DevOps. These artifacts can be published, discovered, pulled, and verified using existing container tooling such as the ORAS CLI (ORAS Project, 2024) and infrastructure (e.g., DockerHub, AWS ECR, Quay.io, Harbor). This introduces a robust, interoperable mechanism for data delivery—supporting structured metadata, referrers, artifact linking, and layered reuse to promote modularity and efficiency.

The remainder of this chapter introduces the relevant components of the OCI specification and presents two architectural strategies for applying OCI to EO workflows:

- Monolithic Packaging Model – Entire datasets or spatial tiles are encapsulated as self-contained, immutable OCI artifacts. This approach is particularly relevant for data package supply chains and is therefore of primary interest in this study.

- Layered Asset Store Model – A modular design in which different data modalities (e.g., optical, radar, annotations) are distributed across distinct layers to support reuse and differential updates.

2.1 The OCI Specification and OCI Artifact Format

The Open Container Initiative (OCI) was established in 2015 under the Linux Foundation to formalize standards for containerized software. Initially focused on container images and runtimes, the initiative has expanded through community-led efforts—most notably the OCI Artifacts project—to accommodate broader use cases, including configuration and policy management, machine learning models, scientific datasets, and general-purpose digital content (Lorenc, 2021).

The OCI specification comprises three core components:

- Image Specification – Defines the structure of artifacts and their metadata.
- Distribution Specification – Governs the protocol for pushing and retrieving artifacts from registries.
- Runtime Specification – Describes how artifacts are executed; less relevant in the context of OCI artifacts used for data rather than software containers.

At the core of an OCI artifact are three structural elements:

- Config Blob – Traditionally used for container runtime parameters, but in EO contexts, it can be repurposed to hold structured metadata such as spatial extent, temporal coverage, data lineage, or sensor type.
- Layers – The actual data payloads, stored as compressed binary blobs and addressed via cryptographic hashes (e.g., SHA-256). Layers may be partitioned by spatial regions, time slices, or thematic dimensions.
- Manifest – The central descriptor linking the config and layers. It specifies digests, media types, and ordering, ensuring that all components of an artifact can be cohesively resolved and retrieved.

```
$ oras manifest fetch localhost:5000/pastis-
2433:sample --format json


{
  "mediaType":
"application/vnd.oci.image.manifest.v1+json",
  "config": {
    "digest": "sha256:6e19...",
    "mediaType":
"application/vnd.oci.image.config.v1+json",
    "size": 3093
  },
  "layers": [
    { "digest": "sha256:7cff...", "mediaType":
"application/vnd.oci.image.layer.v1.tar" },
    { "digest": "sha256:b9af...", "mediaType":
"application/vnd.oci.image.layer.v1.tar" },
    { "digest": "sha256:6f57...", "mediaType":
"application/vnd.oci.image.layer.v1.tar" }
  ],
  "schemaVersion": 2
}
```

Figure 1: Example OCI manifest fetched via ORAS CLI.

Together, these features make OCI registries viable for packaging EO datasets as portable, versioned, and content-addressable artifacts.

2.2 OCI for EO Data Distribution

Several characteristics of OCI align closely with the needs of EO data distribution:

- Layered Storage and Deduplication – Shared components (e.g., base imagery or auxiliary datasets) can be stored once and reused across artifacts. Updates require only the transfer of changed layers.
- Versioning and Traceability – Support for immutable digests and mutable tags enables fine-grained version control. Cryptographic signatures allow provenance and authenticity attestation.
- Infrastructure Ubiquity – OCI registries are widely supported across cloud providers, enterprise platforms, and open-source tools, enabling integration into heterogeneous environments.

As introduced earlier, we identify two emerging design patterns: the self-contained artifact model and the modular, layered asset store. These approaches are not mutually exclusive, but they emphasize different capabilities and trade-offs in data management. Each leverages distinct aspects of the OCI ecosystem to address EO-specific distribution challenges.

2.2.1 Self-Contained OCI Data Packages

Self-contained packaging involves bundling data into monolithic OCI artifacts, where all relevant content is pre-aggregated and structured for consumption as a cohesive unit. In the EO context, this typically means packaging all data for a specific region and time window into a single artifact—often referred to as a patch.

```
├── ANNOTATIONS
│   └── TARGET_10001.npy
├── DATA_S1A
│   └── S1A_10001.npy
├── DATA_S1D
│   └── S1D_10001.npy
├── DATA_S2
│   └── S2_10001.npy
├── DATA_SPOT
│   └── PASTIS_SPOT6_RVB_1M00_2019
│       └── SPOT6_RVB_1M00_2019_10001.tif
└── INSTANCE_ANNOTATIONS
    ├── HEATMAP_10001.npy
    └── INSTANCES_10001.npy
```

Figure 2: Example patch from PASTIS after partitioning.

While OCI layers may still be used internally to partition content (e.g., by file type or modality), the conceptual model is atomic: the artifact is designed to be pulled and consumed in full. Metadata is embedded either in the config object or within the layer files themselves.

This approach aligns well with common EO usage patterns and reflects established practices in high-performance computing (HPC) and machine learning (ML), where datasets and execution environments are often encapsulated together. The ability to retrieve a single, versioned artifact using standard OCI tooling simplifies orchestration and minimizes dependency on external services.

```
$ oras push localhost:5000/pastis-2433:sample \
  --artifact-type application/vnd.whatever.v1+tar \
  --config
config.json:application/vnd.oci.image.config.v1+json \
  10000.tar:application/vnd.oci.image.layer.v1.tar \
  10001.tar:application/vnd.oci.image.layer.v1.tar \
  10002.tar:application/vnd.oci.image.layer.v1.tar

...

$ oras pull localhost:5000/pastis-2433:sample -o /tmp

$ tree /tmp
.
├── config.json
├── 10000.tar
├── 10001.tar
└── 10002.tar
```

Figure 3: Example flow demonstrating publishing (push) and retrieval (pull) of OCI artifacts using the ORAS CLI.

This self-contained model serves as the basis for our experiments. We evaluate it using two intentionally contrasting partitioning strategies applied to the PASTIS dataset:

- PASTIS-2433: The whole PASTIS dataset is divided into 2,433 per-patch subsets. Each patch is packaged into a separate TAR archive and added as a layer in a single OCI artifact. The result is a package with 2,433 layers, each approximately 30–35 MB in size. The config object describes the metadata for each patch.

- PASTIS-t4: The whole PASTIS dataset is divided into four larger tile-based subsets, each covering a distinct spatial region. Each tile is packaged into a separate TAR archive and added as a layer in the OCI artifact. The result is a package with 4 layers, each approximately 15–20 GB in size. The config object contains metadata relevant to each tile.

Given that the OCI standard supports selective layer retrieval, the first partitioning strategy (PASTIS-2433) offers clear advantages in terms of granularity and flexibility compared to the second (PASTIS-t4). Individual patches can be pulled independently, enabling fine-grained access and potentially reducing bandwidth and storage costs when only subsets of the data are required. However, this flexibility comes with trade-offs: if the analysis requires all layers, the overhead of managing a large number of small layers may negatively impact scalability—not only on the registry side but also on the client side (e.g., UI rendering).

That said, the primary goal of this study is not to optimize for a performance metrics, but to demonstrate the viability and flexibility of OCI-based packaging for EO data. We deliberately focus on approaches that align with the ubiquity of existing OCI registries and tooling. Both partitioning strategies were selected to reflect real-world applicability while staying within the practical limits of current OCI infrastructure—and to highlight meaningful contrasts in packaging design.

2.2.2 OCI as a Modular Layered Asset Store

Beyond self-contained packaging, the OCI registry can also function as a modular, general-purpose content store, more closely aligned with how data is structured in live and mutable data streams. Interestingly, even platforms like Hugging Face Datasets and EOTDL, which provide curated and versioned snapshots, often retain an internal structure that reflects their streaming origins—organized by product source, timestamps, or other data modalities.

```
$ tree
.
├── ANNOTATIONS
│   ├── ParcelIDs_10000.npy
│   ├── ParcelIDs_10001.npy
│   ├── ParcelIDs_10002.npy
│   ├── TARGET_10000.npy
│   ├── TARGET_10001.npy
│   ├── TARGET_10002.npy
 ...
├── DATA_S1A
│   ├── S1A_10000.npy
│   ├── S1A_10001.npy
│   ├── S1A_10002.npy
 ...
├── DATA_S1D
│   ├── S1D_10000.npy
│   ├── S1D_10001.npy
│   ├── S1D_10002.npy
 ...
├── DATA_S2
│   ├── S2_10000.npy
│   ├── S2_10001.npy
│   ├── S2_10002.npy
 ...
├── DATA_SPOT
│   └── PASTIS_SPOT6_RVB_1M00_2019
│       ├── SPOT6_RVB_1M00_2019_10000.tif
│       ├── SPOT6_RVB_1M00_2019_10001.tif
│       ├── SPOT6_RVB_1M00_2019_10002.tif
 ...
├── INSTANCE_ANNOTATIONS
│   ├── HEATMAP_10000.npy
│   ├── HEATMAP_10001.npy
│   ├── HEATMAP_10002.npy
│   ├── INSTANCES_10000.npy
│   ├── INSTANCES_10001.npy
│   ├── INSTANCES_10002.npy
│   ...
```

Figure 4: Unpartitioned version of the PASTIS dataset.

This design follows the natural grouping principle of keeping data from the same source together. It supports the incremental addition of new data to each source segment as it becomes available, thereby promoting extensibility and updatability. These source segments can even be packaged as separate OCI artifacts, each with its own manifest and index. The OCI Referrers API allows these individual artifacts to be linked into composite datasets, enabling the registry to maintain coherence across related components.

However, this approach comes at the cost of increased packaging complexity and may require more sophisticated client logic to interpret and resolve referrers—overhead that is unnecessary in cases involving immutable, packaged datasets, which remain the primary focus of this study.

2.2.3 Industry Trends

Several domains have begun adopting OCI registries for structured, versioned, and modular data distribution. The bioinformatics community has explored OCI for packaging genomic references and alignment indices, where modularity and reproducibility are key (Davis et al., 2021). In high-performance computing, projects using Apptainer and Singularity have pushed scientific data containers to OCI registries, often separating simulation input/output from runtime environments (Apptainer Project, 2023). Similarly, ML training pipelines are starting to use registries to manage large datasets and model checkpoints as standalone artifacts, benefiting from OCI's support for layering, metadata, and reuse (Lorenc, 2023).

In support of this direction, Docker Inc. recently introduced support for non-container artifact types on Docker Hub, including a dedicated model type for machine learning (ML) payloads (Docker Inc., 2022). These artifacts feature structured metadata, rich UI integration, and type-specific search capabilities—validating the layered registry model. Earth Observation (EO) datasets could adopt similar conventions by registering artifact types such as earth-observation.

More recently, Docker introduced the Docker Model Runner, which allows ML models to be stored in OCI registries and directly executed for inference—blurring the line between packaging and serving (Docker Inc., 2024). This approach mirrors a concept already adopted by KServe, a Kubernetes-native model serving platform that uses OCI artifacts to package and manage versioned models, enhancing reproducibility and traceability in machine learning workflows (KServe Documentation, n.d.).

2.2.4 Insurance Industry Agricultural Use Case

To illustrate the rationale behind the proposed data packaging approach, we briefly outline a real-world scenario inspired by actual requirements in the agricultural insurance sector. In this case, an insurance company needs to combine various geospatial data sources to support field operations such as crop monitoring, damage assessment, and irrigation analysis.

The company operates an internal, modular data lake, but delivers curated, attestable data packages to field agents using an OCI artifacts. This layer assembles relevant content into optimized artifacts for efficient, on-demand consumption.

- Data Assembly: The core dataset is following a similar structure as PASTIS, containing satellite timeseries as well as VHR satellite imagery, extended with

additional layers such as meteorological data, soil moisture indices, and irrigation-related information. Data is spatially and temporally aligned during preprocessing, with reprojection or resampling applied as needed. The system regularly synchronizes with the inputs from the data lake.

- Packaging & Metadata: The processed data is structured into region-specific layers (comparable to the patching concept described earlier). A corresponding config blob includes metadata such as bounding boxes, acquisition dates, time series durations, and provenance including automated quality masks and attestations. Packages are published to be an OCI registry.
- Field Agent Usage: Field agents download the relevant packages for the areas they plan to visit upfront—typically onto mobile edge devices, even in offline regions with limited or no internet connectivity. These devices support exploratory data tools and on-device ML inference, enabling real-time analysis directly in the field. All actions—including the models used, predictions made, and associated data attestations—are logged for auditability and compliance.
- Model Review and Benchmarking: Internal teams (Analysts, Data Scientists) reuse the same OCI artifacts to review model performance, conduct benchmarking, and maintain consistency across development and deployment. Metadata and manifests provide the necessary hooks for reproducibility and traceability.

This use case demonstrates how OCI artifacts can enhance data governance, reproducibility, and operational efficiency in EO-driven workflows. By adopting OCI's modular and verifiable packaging model, organizations can streamline data delivery, ensure data integrity, and bridge the gap between centralized processing and decentralized consumption in the field.

3. Methodology

A central methodological focus of this study is to evaluate how Earth Observation (EO)–derived, use-case-specific datasets—typically the result of fusion, harmonization, and preprocessing pipelines—can be structured, packaged, and distributed effectively as OCI artifacts.

Unlike conventional approaches that store raw EO data in generic data lakes or expose them via APIs optimized for exploratory access—usually involving an open-ended, two-step process of search and retrieval—this work concentrates on the delivery of ready-to-use, tightly scoped data bundles. These curated packages are designed to support specific downstream tasks, such as on-device inference, model training, regulatory compliance auditing, or manual field assessments on remote edge devices. The goal is not to enable search and browsing, but to ensure that the correct partition of pre-validated data is directly applicable to a targeted operational task.

While OCI artifacts are not inherently file-based, they encapsulate file-based content in a structure that is both modular and familiar. The use of immutable layers to store data and a config blob to store metadata creates a layout that closely resembles traditional file-based storage. This is a significant advantage, as it aligns with the mental models of users, machine learning frameworks such as PyTorch and TensorFlow, and many legacy geospatial tools and libraries that assume file-based input. Thus, although OCI artifacts are formally structured as object-based packages, their ability to encapsulate and present file-based content makes them both practical and interoperable across a wide range of EO applications. This dual nature bridges the gap between modern object storage paradigms and the file-based expectations of existing tools and workflows.

The broader implication of this methodology is that domain-specific packaging standards can—and should—emerge from practical, real-world usage. While the OCI specification provides a robust structural foundation, achieving true interoperability will require community-driven conventions, including consistent field naming, labeling practices, and annotation schemas.

Establishing such conventions begins with pragmatic experimentation. This study aims to lay that groundwork—demonstrating viable models that can be refined and expanded through collaborative iteration.

3.1 Structuring EO-Derived Data Products as OCI Artifacts

The individual layers of an OCI artifact correspond to logical partitions of the dataset. Each layer represents the actual data for a given partition—whether defined spatially, temporally, or thematically—and is stored as a TAR archive, optionally compressed (e.g., using gzip or zstd). These archives contain all content relevant to that partition, ensuring self-sufficiency.

This paper does not focus on the internal data formats of each partition; instead, it assumes that all bytes of a partition are needed once the layer is pulled. As such, partial access or streaming within a layer is not a primary concern. However, the efficiency of compression remains relevant: specifically, whether the savings in transfer size justify the computational cost of decompression at the edge or client side. The choice of compression algorithm can significantly impact usability depending on the execution environment—whether in the cloud, on local clusters, or at remote edge devices. Even after the data is pulled to a target environment, the chosen format and compression scheme determine whether additional extraction steps are required or if the data can be consumed immediately. For example, simple TAR archives may be preferable for lightweight, offline processing, as they avoid the overhead associated with more complex or nested formats. In such contexts, selecting a compression method that balances size reduction with low extraction latency becomes crucial. From the OCI registry's perspective, both the number of layers and the size of individual layers are relevant factors.

While the OCI specification defines a consistent structure for how layers are represented—including media types such as:

- application/vnd.oci.image.layer.v1.tar
- application/vnd.oci.image.layer.v1.tar+gzip
- application/vnd.oci.image.layer.v1.tar+zstd

—it does not enforce strict standardization of the internal formats used within those layers. Instead, these aspects are governed by convention and shaped by domain-specific requirements, allowing flexibility in how content is structured and interpreted within each layer (OCI Image Specification, n.d.).

OCI tooling—such as Docker, ORAS CLI—natively supports these formats. With the introduction of OCI Artifacts in version 1.1, it is now also possible to define custom media types at both the artifact and layer level. One could define types such as:

- application/vnd.org.geotiff.layer.v1+tar
- application/vnd.eo.timeseries+zarr

to signal domain semantics and expectations. However, most OCI clients do not treat these media types differently unless explicitly configured to do so. As such, they serve primarily as descriptive metadata rather than functional directives for behavior or processing. In this study, we adhered to the existing, widely supported layer media types to ensure broad compatibility across registries and tools.

Next to the layers sits the config object, intended for descriptive, domain-specific metadata. It is not involved in the layout or retrieval of the artifact but provides a flexible space for users to encode meaningful information about the artifact's contents. The config blob is required (OCI Image Specification, 2023) and must be a valid JSON object with the media type

- application/vnd.oci.image.config.v1+json

In this study, we chose to leverage the config object as a custom metadata schema that enriches the artifact beyond its technical structure. Specifically, we:

- Included a list of all dataset partitions (each corresponding to a layer),
- Attached relevant metadata for each partition (e.g., spatial boundaries, temporal coverage), and
- Embedded the digest of each TAR file directly in the config, matching the corresponding entry in the manifest.

This dual declaration—linking layers by digest in both the manifest and the config—serves multiple purposes. It enhances data integrity, supports verifiability, and enables domain-specific clients to validate or inspect the artifact structure independently of OCI tooling. In this sense, the manifest acts as a technical table of contents, while the config serves as a semantic index or annotation layer—intended for clients, researchers, and operational workflows.

For EO-specific applications, this approach aligns well with existing standards such as STAC and related OGC metadata models, which define common field names and structures for describing geospatial assets. Mapping those conventions into, or even directly using, a STAC Item or OGC record as the OCI config allows artifacts to remain compatible with established metadata expectations.

A practical limitation of the current OCI specification is that only a single config blob can be attached to an artifact, and it must be a JSON object with a defined media type (OCI Image Specification, 2023). For datasets involving thousands of partitions or fine-grained provenance, encoding all metadata into a single config can become cumbersome or inefficient.

At the same time, this raises the broader question of whether such a use case still constitutes a well-scoped data package—or whether it would be more appropriate to treat it as a modular asset store (cf. Section 2.2.2), where metadata catalogs must support mutability and implementation should follow different strategies.

3.2 Evaluation Setup and used OCI Registries

To assess practical suitability, we conducted a comparative evaluation across a representative selection of OCI-compatible registry backends. These include public, enterprise-managed, and self-hosted services:

- Docker Hub – docker.io/versioneer (Docker Inc., 2025)
- Quay.io – quay.io/versioneer-inc (Red Hat, 2025)
- Harbor (hosted by OVHCloud) -###.c1.de1.container-registry.ovh.net/versioneer (Harbor Project, n.d.)
- Amazon Elastic Container Registry –###.dkr.ecr.eu-central-1.amazonaws.com/versioneer (Amazon Web Services, 2025)
- Zot Registry - localhost:5000 (Zot Registry, n.d.)

As benchmark artifacts, we used three OCI-compatible data packages derived from the PASTIS dataset (cf. Section 2.2.1), designed to reflect diverse packaging strategies and workload characteristics:

- pastis-2433:full - high-partitioned artifact comprising 2,433 individual layers (~30-35 MB each)
- pastis-t4:full - low-partitioned artifact with 4 large layers (~15–20 GB each)
- pastis-2433:sample - minimal variant of pastis-2433:full, including only the first three layers, serves both as a lightweight smoke test and as a reproducible reference published in the companion GitHub repository

All artifacts were created and pushed using the ORAS CLI, relying on standard OCI operations without registry-specific tuning or enhancements. This approach reflects the typical usage pattern expected emphasizing interoperability and out-of-the-box compatibility. Uploads were performed using consistent commands, and optional concurrency flags were applied uniformly where supported. This controlled setup ensures a fair comparison of registry behavior under realistic but standardized conditions.

3.3 Evaluation Dimensions for OCI Registry Suitability

To assess the registry-side behavior relevant to data distribution, we define a focused set of evaluation dimensions derived from the characteristics of the benchmark artifacts. Each dimension targets a specific feature or constraint that may influence the packaging, transfer, or usability of OCI artifacts. Wherever available, we relied on official registry documentation to establish expected behavior. Direct testing was conducted when behavior was undocumented, implementation-specific, or expected to vary.

- Size-related constraints are a fundamental consideration, as registries may enforce limits on individual layer sizes or overall artifact size. Given the presence of large blobs as well as many layers in our benchmark artifacts, we verified registry behavior through empirical uploads across all cases.
- Storage deduplication was evaluated by examining whether shared layers between artifact variants (e.g., full and sample) were re-uploaded or recognized and skipped, and how this influenced reported storage consumption. While most OCI registries implement content-addressable storage, we verified this behavior experimentally to confirm effective layer reuse.
- Support for selective layer access via blob digests was recorded, as all registries conforming to the OCI specification expose blob-level APIs. Although not a differentiating feature in practice, we verified this behavior experimentally to confirm that individual

layers could be retrieved independently of the full artifact.
- Resumability of uploads and downloads was evaluated insofar as it could be observed through client tooling. Where documentation was insufficient, we conducted targeted tests to assess whether interrupted sessions could be resumed without restarting the entire transfer.
- Support for the OCI Referrers API was determined from registry announcements and specifications. Since rollout is ongoing across platforms, we limited testing to confirming acceptance of referrer uploads and querying referrers where supported.
- Finally, we noted whether registries preserve custom annotations and tolerate non-standard media types. These features are part of the OCI specification, and we assumed compliance unless documentation or prior reports suggested limitations. Where feasible, we confirmed manifest integrity after upload.

This set of dimensions offers a practical framework to characterize registry behavior with respect to layered, large-scale, and metadata-rich artifacts, while balancing empirical testing with documentation-based inference.

4. Results

The complete evaluation benchmark—including setup, scripts, and discussion of results—has been published in the research section of our website (Versioneer, 2025).

The following summarizes observed registry behavior across a focused set of evaluation dimensions tailored to our benchmark artifacts. Findings combine direct testing with documentation analysis and highlight only behavior with practical impact.

- Docker Hub: Empirical testing confirmed that Docker Hub accepted artifacts containing both many small and a few large layers. However, reliability decreased for multi-gigabyte layers due to throttling. Shared layers were correctly deduplicated across tags. Blob-level access worked as expected, and resumable uploads/downloads were supported via client retries. OCI referrers are not yet supported. While custom media types and annotations were preserved, the Docker Hub UI does not expose them.
- Quay.io: The managed Quay service reliably handled artifacts across all tested size and layer count configurations. It demonstrated effective deduplication, supported blob-level access and partial downloads, and allowed upload/download resumption within protocol limits. The Referrers API is fully supported, and both annotations and custom media types were preserved without issue.
- Harbor (hosted by OVHCloud): The private Harbor instance accepted all artifacts and supported high layer counts. Deduplication and blob access functioned reliably, and resumable transfers were validated. OCI referrers were supported, and custom metadata was retained and visible in the Harbor UI.
- Amazon Elastic Container Registry: The private AWS instance supported large layer sizes but enforced a hard limit of 500 layers per artifact, which restricts its use for highly partitioned datasets. This limitation is documented and not configurable. Deduplication was effective across repositories, and blob-level access was confirmed via API. Upload/download resumption was supported through client logic. OCI referrers are supported, and while custom metadata is retained, it is not displayed in the AWS Console UI.
- Zot Registry (local): The local Zot Registry handled both artifact variants without issues related to size or layer count. Layer reuse worked as expected, and HTTP range requests enabled blob-level access. Transfer resumption is supported at the protocol level. Annotations and custom media types were preserved.

5. Conclusion and Acknowledgements

Our evaluation results strongly indicate that today's landscape of OCI registries—spanning public platforms, managed enterprise services, and open-source deployments—is well-equipped to handle data artifacts of varying granularity and size. This makes them readily usable for Earth Observation (EO) data packages in the range of several dozen gigabytes without requiring substantial adaptation or customization. Core features such as deduplication, blob-level access, resumability, and metadata preservation are reliably supported, providing a robust foundation for scalable and interoperable data distribution.

While our focus was on registry-side behavior, it is clear that the broader OCI ecosystem—including tools for building and managing artifacts—is equally critical. Although the ORAS CLI served as our primary interface, tools such as Docker, Podman, Skopeo, Singularity, and emerging domain-specific clients are becoming increasingly relevant. Their suitability for EO-specific workflows—or the potential need for purpose-built tooling—remains an open area for further investigation.

Standardization is another critical area of ongoing work, particularly around domain-specific media types, annotation keys, and metadata conventions. This challenge is not unique to EO. The machine learning community, for example, is actively developing OCI-based packaging strategies for ML models to streamline the process from model packaging to inference results. Such developments reflect a broader momentum toward formalizing domain-aware artifact specifications atop the OCI foundation.

Beyond the distribution of immutable data packages within data product supply chains, OCI's foundational principles—layered storage, content addressing, and manifest-driven composition—make it a compelling candidate for general-purpose data asset management. As columnar and chunked formats gain traction and high-performance analytical tools become mainstream, OCI registries could evolve into intelligent, versioned storage backends for next-generation data workflows.

In a general-purpose asset store, mutability is an inherent requirement—essential for managing metadata that evolves over time, reflects ingestion progress, or supports dynamic indexing. A key enabler of this model is support for partial access, particularly through HTTP range requests. These allow clients to retrieve only the necessary byte ranges from large blobs—especially important for formats such as COG, Zarr, or GeoParquet. In such contexts, partial access is not merely a performance optimization but a structural necessity. Storing metadata in queryable formats like GeoParquet enables efficient remote access without requiring full file downloads. Building on this, open table formats and versioned indexing schemes offer promising mechanisms for supporting scalable, dynamic metadata layers atop OCI-based infrastructure.

We gratefully acknowledge the vibrant open-source geospatial community—particularly initiatives like Pangeo and Cloud-Native Geospatial, and pioneering teams such as Development

Seed and Earthmover—for their continued contributions that drive innovation and make cutting-edge solutions widely accessible. We are also fortunate to collaborate with outstanding partners and colleagues at Versioneer and EOX, whose support fosters a creative environment and enables us to explore bold, cloud-native approaches to Earth Observation data infrastructure.

Special thanks to OVHCloud for supporting this research through their Starter Program by providing cloud credits.

References

Amazon Web Services, 2025: Amazon Elastic Container Registry (ECR). https://docs.aws.amazon.com/ecr/ (15 May 2025).

Anghelea, A., Smith, G., Meissl, S., & Achtsnit, S., 2024: EarthCODE – ESA's Earth Science Collaborative Open Development Environment. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4-2024, 19–25. doi:10.5194/isprs-archives-XLVIII-4-2024-19-2024.

Apptainer Project, 2023: Singularity and Docker Compatibility. https://apptainer.org/user-docs/master/singularity_and_docker.html (15 May 2025).

Docker Inc., 2022: Announcing Docker Hub OCI Artifacts Support. https://www.docker.com/blog/announcing-docker-hub-oci-artifacts-support/ (15 May 2025).

Docker Inc., 2024: Introducing Docker Model Runner – A Simpler Way to Serve Models. https://www.docker.com/blog/introducing-docker-model-runner/ (15 May 2025).

Docker Inc., 2025: Docker Hub – Cloud-based OCI Registry. https://hub.docker.com (15 May 2025).

EOTDL, n.d.: Earth Observation Training Data Library. https://eotdl.com (15 May 2025).

Food and Agriculture Organization (FAO), n.d.: Use of Earth Observation Data. https://www.fao.org/in-action/eostat (15 May 2025).

Harbor, n.d.: Open Source OCI Registry. CNCF project. https://goharbor.io (15 May 2025).
Docker

Hugging Face, n.d.: Hugging Face Datasets. https://huggingface.co/datasets (15 May 2025).

IPFS, n.d.: InterPlanetary File System. https://ipfs.tech (15 May 2025).

KServe Documentation, n.d.: Serving Models with OCI Images. https://kserve.github.io/website/latest/modelserving/storage/oci/ (15 May 2025).

Lorenc, D., 2021: OCI Artifacts Explained. Are they real? Kind of! https://dlorenc.medium.com/oci-artifacts-explained-8f4a77945c13 (15 May 2025).

Lorenc, D., 2023: OCI as a Standard for ML Artifact Storage and Retrieval. https://www.youtube.com/watch?v=hGM5KQLzbYc (15 May 2025).

OCI Image Specification, n.d.: Media Types. https://github.com/opencontainers/image-spec/blob/main/media-types.md (15 May 2025).

Open Container Initiative (OCI), 2023: About the Open Container Initiative – OCI project overview. https://opencontainers.org (15 May 2025).

ORAS Project, 2024: OCI Registry As Storage (ORAS) CLI Tool. https://oras.land (15 May 2025).

Red Hat, 2025: Quay.io – OCI-Compatible Registry Service. https://quay.io (15 May 2025).

Rolla, A., 2025: Satellite-Aided Disaster Response. *AGU Advances*, 6, e2024AV001395. doi:10.1029/2024AV001395.

Sainte Fare Garnot, V., & Landrieu, L., 2021: Panoptic Segmentation of Satellite Image Time Series with Convolutional Temporal Attention Networks. https://arxiv.org/abs/2107.07933 (15 May 2025).

STAC, 2023: SpatioTemporal Asset Catalog Specification. https://stacspec.org (15 May 2025).

Turner, P., 2024: Orchestrating Bioinformatics Workflows Across a Heterogeneous Toolset with Flyte. *SciPy 2024 Proceedings*. https://proceedings.scipy.org/articles/DDJJ4932.pdf (15 May 2025).

Versioneer, 2025: Research area. https://research.versioneer.at/paper-oci-supply-chain (15 May 2025).

World Meteorological Organization (WMO), 2024: State of Climate Services 2024. https://wmo.int/publication-series/2024-state-of-climate-services (15 May 2025).

Zot Registry, n.d.: Open Source OCI-Compatible Registry. https://github.com/project-zot/zot (15 May 2025).