



Leibniz-Institut für  
Astrophysik Potsdam

# **S3-Enabled Reproducible Workflows**

## **Bridging REANA, DRPHub, and AI-Driven Data Analysis**

Enrico Tom Leonhard Strauß & Dr. Arman Khalatyan / AG2026 - e-Science Workgroup AG Meeting / 8. September 2026

# Motivation: Storage Is the Reproducibility Bottleneck

- **Reproducible analysis needs four things:** code, environment, data, execution record — the data leg is the weakest.
- **Workflows are coupled to the cluster:** POSIX shared filesystems (NFS, CephFS) tie analyses to one infrastructure; REANA currently requires a shared filesystem for workspaces.
- **Shared filesystems limit scale:** TB–PB survey data, portability across on-prem and cloud, and cross-institution sharing.
- **What is missing:** a standard access API, built-in versioning, immutability, lifecycle policies, and addressable data objects.
- **The S3 model is the de facto standard:** HTTP-native API, object versioning, lifecycle, preassigned URLs - identical on MinIO, AWS, RustFS, VersityGW

# REANA: From Shared Filesystem to Native S3

- **Today:** workspaces on CephFS/NFS; inputs and outputs declared in reana.yaml as files and directories.
- **Direction:** S3-compatible staging (rclone-based) so workflows address s3:// URIs directly.
- **Portability:** the same analysis runs on-prem, in cloud, or beside HPC — no remounting.
- **Provenance:** content-addressed inputs with Etags (metadata); versioned, lockable objects (bucket-level object lock).

## reana.yaml — the concept

inputs:

files:

# before

- data/catalog.fits

# after

- /data/s3/mysh/catalog.fits

outputs:

files:

- results/\*.parquet

# after

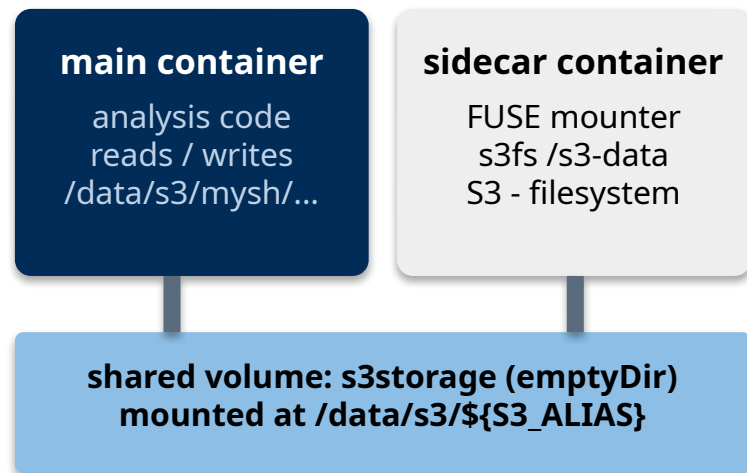
- /data/s3/mysh/results/

# Implementation: S3 Mount via Sidecar FUSE

[reanahub/reana-workflow-controller#671](#)

- **Alias-based mounts:** each user bucket appears at `/data/s3/${S3_ALIAS}` (e.g. `/data/s3/mysh`) - conflict-free, no global config.
- **Credentials via secrets:** `S3_TO_LOCAL_${S3_ALIAS}_*` variables — registered with reana-client or synced from DRPHub.
- **Sidecar FUSE mounter:** `s3fs /s3-data` mounts the bucket into a shared emptyDir volume visible to the job container.
- **Code changes:** `k8s.py`, `workflow_run_manager.py` in `reana-workflow-controller / job-controller` and new `reana-datastore-s3fs` repository.
- **Optional region:** endpoint-aware, e.g. 'de-ost-aip' (AIP). RFC in the PUNCH4NFDI community.

Pod (workflow job)



# Configuration: Required Secrets and Setup

Register as REANA secrets (via DRPHub sync or reana-client)

| Variable                                         | Description     | Example    |
|--------------------------------------------------|-----------------|------------|
| <code>S3_TO_LOCAL_\${S3_ALIAS}_ALIAS</code>      | User S3 alias   | mysh       |
| <code>S3_TO_LOCAL_\${S3_ALIAS}_ACCESS_KEY</code> | S3 access key   | XXXX       |
| <code>S3_TO_LOCAL_\${S3_ALIAS}_BUCKET</code>     | Target bucket   | my-bucket  |
| <code>S3_TO_LOCAL_\${S3_ALIAS}_HOST</code>       | S3 endpoint     | s3.aip.de  |
| <code>S3_TO_LOCAL_\${S3_ALIAS}_SECRET_KEY</code> | S3 secret key   | YYYY       |
| <code>S3_TO_LOCAL_\${S3_ALIAS}_REGION</code>     | Optional region | de-ost-aip |

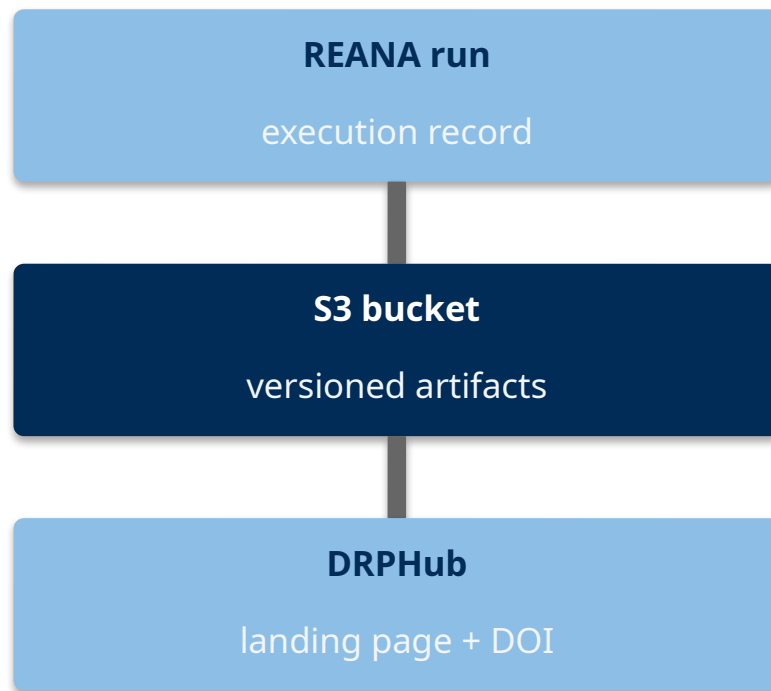
## Setup example

```
export S3_ALIAS=mysh
```

```
reana-client secrets-add \  
  --env S3_TO_LOCAL_${S3_ALIAS}_ALIAS=$S3_ALIAS  
reana-client secrets-add \  
  --env S3_TO_LOCAL_${S3_ALIAS}_ACCESS_KEY=XXXX  
reana-client secrets-add \  
  --env S3_TO_LOCAL_${S3_ALIAS}_BUCKET=mybucket  
reana-client secrets-add \  
  --env S3_TO_LOCAL_${S3_ALIAS}_HOST=s3.aip.de  
reana-client secrets-add \  
  --env S3_TO_LOCAL_${S3_ALIAS}_SECRET_KEY=YYYY  
reana-client secrets-add \  
  --env S3_TO_LOCAL_${S3_ALIAS}_REGION="de-ost-aip"
```

# DRPHub: S3 as the Artifact and Provenance Layer

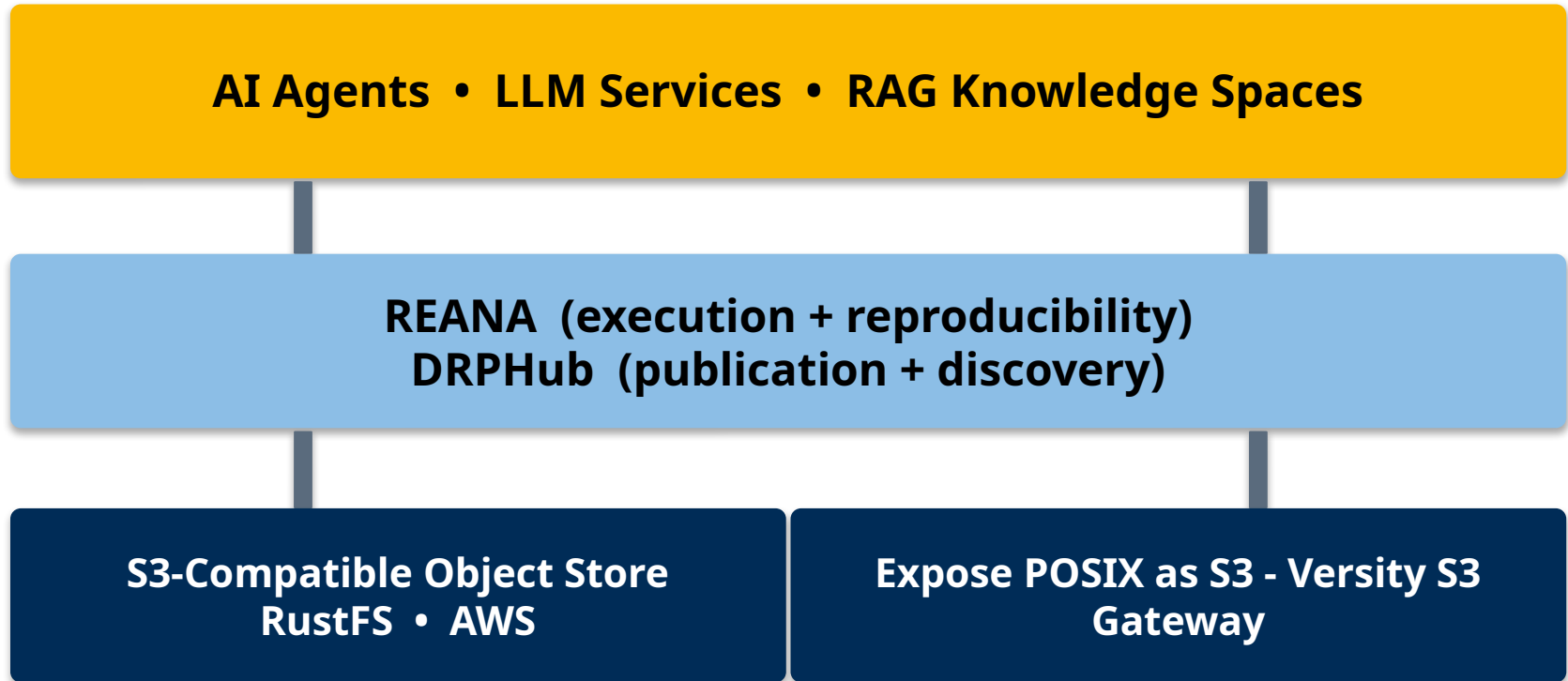
- **Publication layer:** workflow artifacts, intermediate products, and releases become first-class versioned S3 objects.
- **Sharing:** preassingend-URL access without account coupling; checksummed, citable, persistent artifacts.
- **Lifecycle:** policy-driven separation of scratch vs. archival storage.
- **Provenance chain:** code (Git) → environment (container) → data (S3 object + version ID + ETag) → execution (REANA run record).
- **Result:** essentially FAIR compliance - findable, accessible, interoperable, reusable by design.



## Why AI Pipelines Need S3

- **Already the ML standard:** training data, checkpoints, embeddings and feature stores, and model artifacts live in S3 in production ML stacks.
- **Agents and LLMs need:** cheap, parallel, ranged access to large datasets - object storage without POSIX overhead; even AWS now bridges S3 to file-system semantics for ML jobs.
- **Astronomy case:** agents orchestrating survey-scale analysis (multi-TB images, catalogs, time series) over a shared bucket space of versioned intermediate products and embeddings.
- **Agentic workflows:** generate reproducible DAGs - but they need deterministic, addressable storage underneath.
- **Bottom line:** one storage layer serves humans, workflows, and agents.

# Converged Architecture: One Storage Contract



# Roadmap and Open Questions

## Near term

S3 staging for REANA inputs/outputs • sidecar FUSE mounts (RFC #671) • DRPHub artifact buckets

## Mid term

S3-backed workspaces • agent-readable object metadata for automated discovery

## Open questions

quotas and accounting for multi-user buckets • self-hosted vs. institutional Ceph / RustFS • GPU-data locality

## Takeaways

**S3 is the missing portability layer:** reproducible workflows decouple analysis from infrastructure.

**Our sidecar FUSE mount makes it real:** buckets appear as plain /data/s3/ paths inside REANA jobs, credentials isolated per alias.

**The same S3 layer serves AI agents:** one investment, two returns (one storage layer serves humans, workflows, and agents)

**Thank you - questions?**