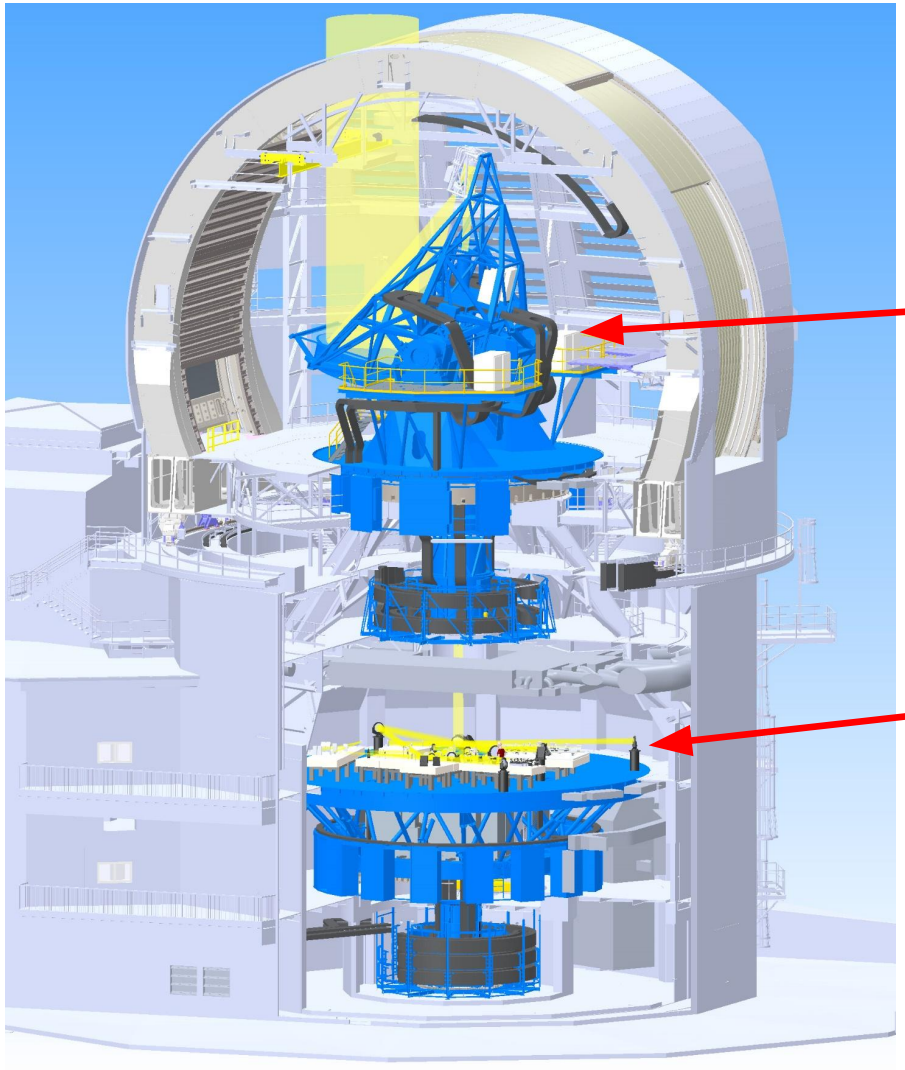
DKIST Data Management

Robert Tawa

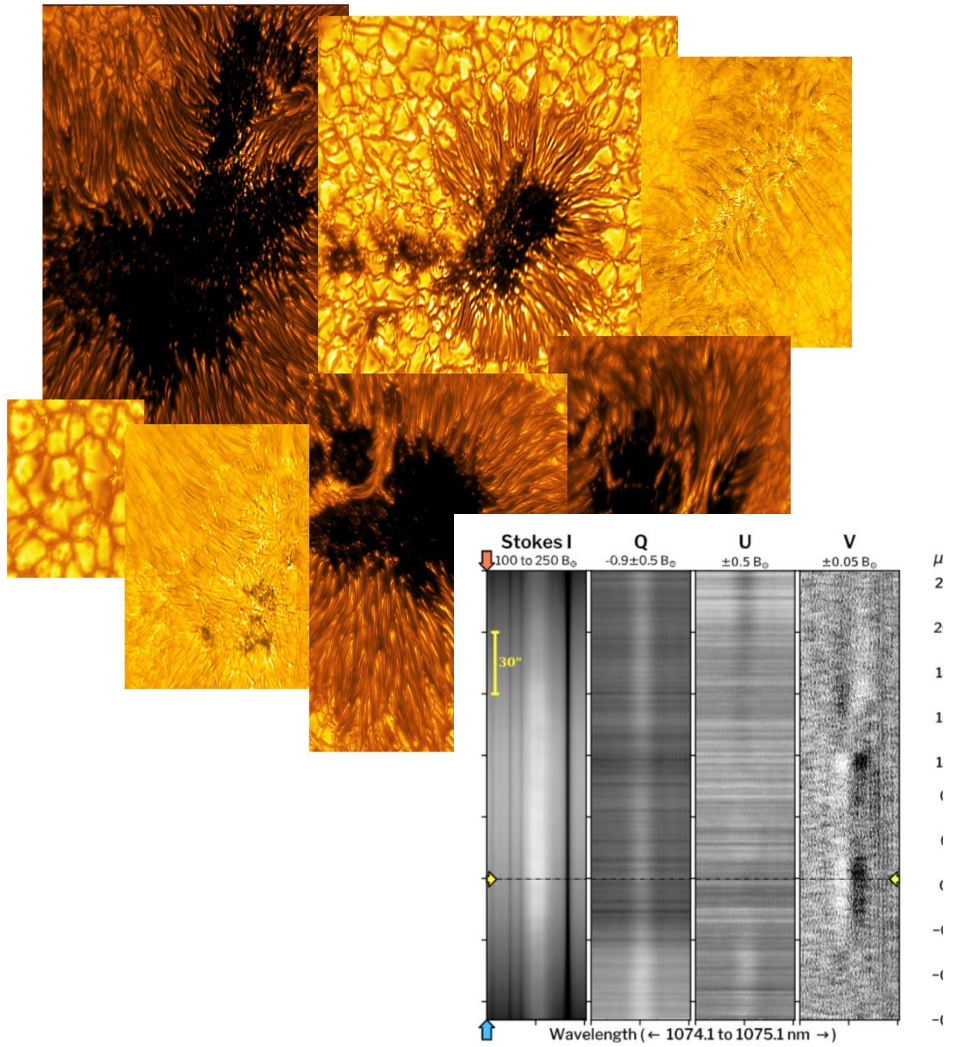
Project Overview



- DKIST is the most advanced and technically complex solar observatory in the world.
  - 4 m aperture with wavefront correction system (AO) allows for diffraction-limited observations down to 20 km resolution
  - Off-axis design and coronal skies (above Haleakala, Maui, HI) allows for sensitive polarimetric measurements of the corona
  - 4 (soon to be 5) optical and IR instruments within an environmentally controlled, rotating Coudé laboratory

DKIST Instruments

- ***Visible Spectro-polarimeter*** (ViSP)
  - Slit Spectrograph
- ***Visible Broadband Imager*** (VBI)
  - Takes high spatial resolution images of the solar surface and low atmosphere.
- ***Cryogenic Near Infrared Spectro-polarimeter*** (Cryo-NIRSP)
  - Designed to study the solar atmosphere at infrared wavelengths, focusing primarily on the solar corona
- ***Diffraction Limited Near Infrared Spectro-polarimeter (DL-NIRSP)***
  - Diffraction grating based integral field spectrograph
  - Measures 2-dimensional spatial and spectral information simultaneously
- ***Visible Tunable Filter*** (VTF)
  - Imaging spectro-polarimeter
  - Takes very high spatial resolution images while scanning in wavelength through a spectrum line at high speed



Introduction and Scope

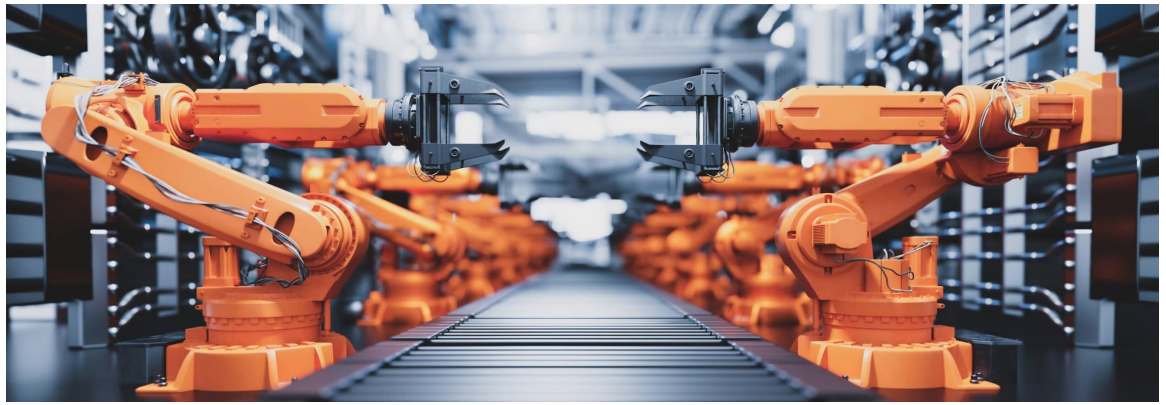
- DKIST represents a new paradigm for for solar science
  - Service-mode (dynamically-scheduled queue) observing based on accepted User proposals
    - Improved efficiency over previous access (PI) mode observing where the PI gets exclusive access to the telescope for some period of time
  - Large volumes of data (>10 TB/day with VTF) require automated data processing and dissemination to the community

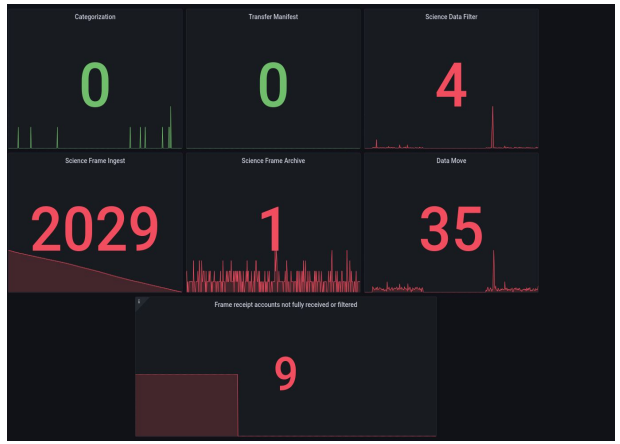
Data Center Context

- Data Center (DC) team is responsible to receive, process, store, and curate Inouye solar data - large volume (PB/year) for 40 yrs, large file volumes (Millions/yr)
- Telescope observes during the day (HI time) overnight batch transfer of observe and calibration data

We have a small team (10) so…

- Automation and system visibility are key
  - The DC designed for maximum automation
    - Automated data movement & pipelines
    - Automated monitoring with notifications
    - Automated testing
    - Automated system setup and orchestration
- The DC has implemented system wide monitoring and alerting solutions
  - Quickly find small issues before they become big issues
  - Real time assessment of the performance every part of the system so as scale/fix as necessary





Data Context

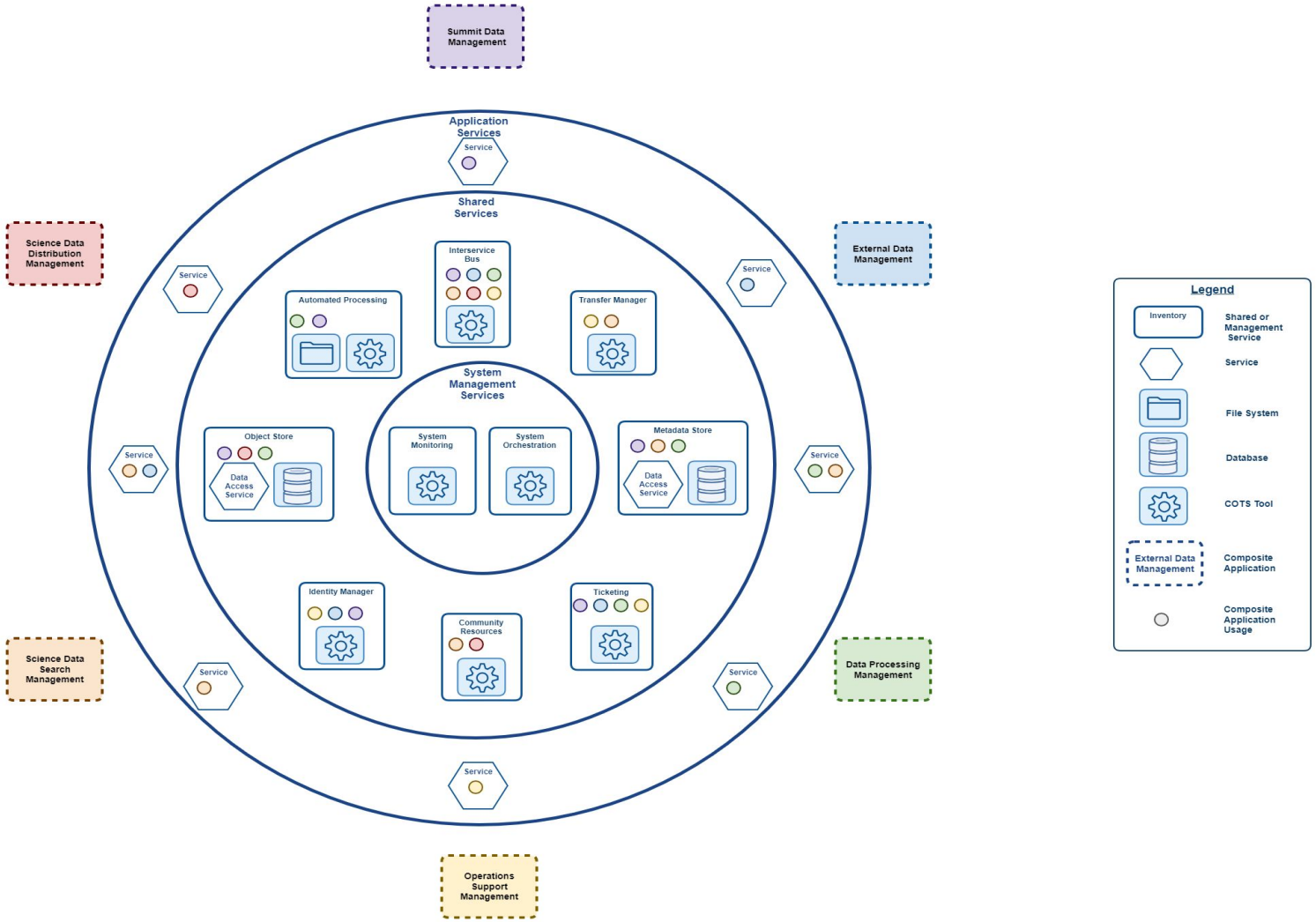
- The DKIST DC must be able to
  - Store the data
  - Augment the data with more data and metadata
- DKIST DC cannot lose L0 (raw) data
  - DC periodically audits its data holdings to assess whether any data loss is occurring
- The DKIST must enable searching/filtering the data on a number (currently 19) of axes (instrument, spectral, wavelength, time, etc)

Architecture

Architecture “ilities”

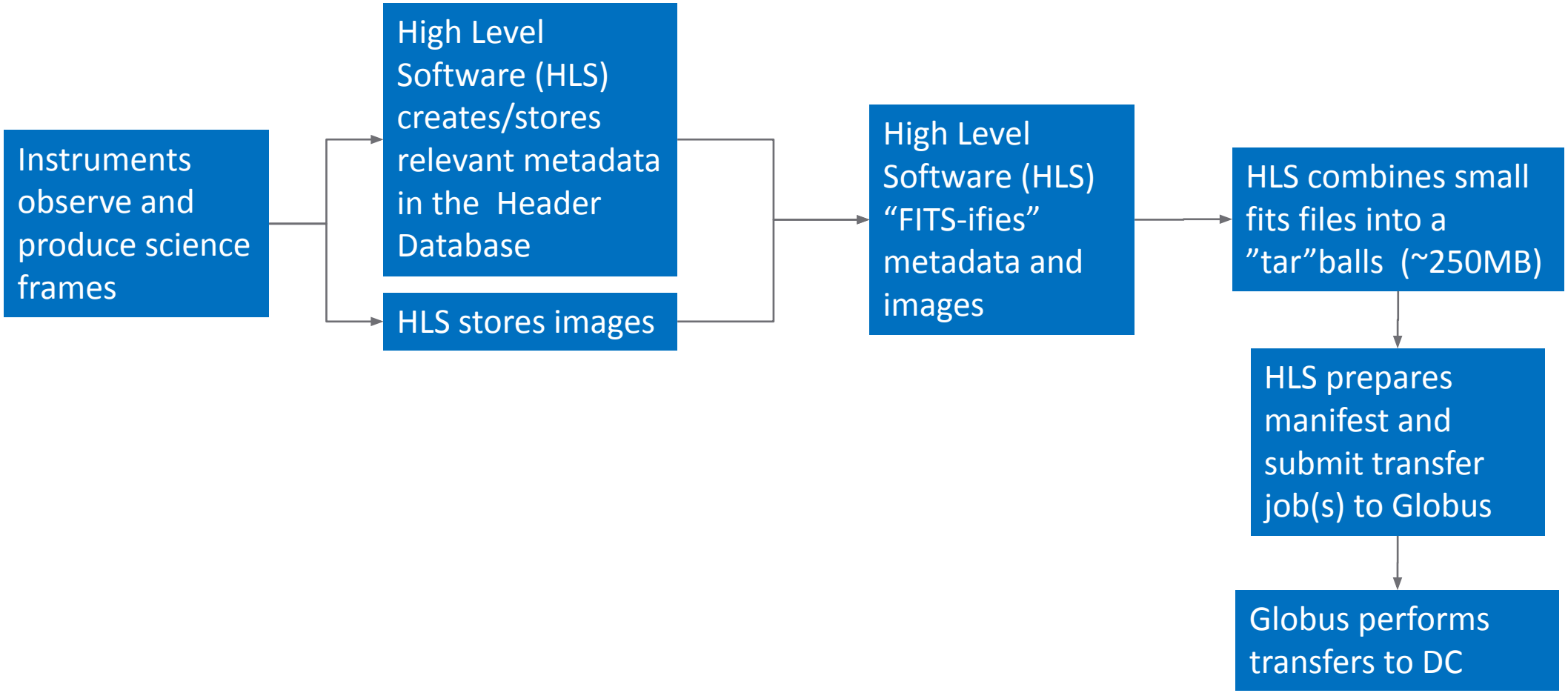
Composite Application	Scalable	Maintainable	Extensible	Reliable	Flexible	Usable	Accessible	Testable	Affordable
Summit Data Management	1	1	0	1	0	0	1	1	1
External Data Management	0	1	0	1	0	0	1	1	1
Science Data Processing Management	1	1	1	1	1	0	0	1	1
Science Data Search Management	0	1	1	1	0	1	0	1	1
Science Data Distribution	1	1	0	1	0	1	0	1	1
Operations Management	0	1	0	1	0	0	0	0	1

DKIST DC software architecture



Data Ingest

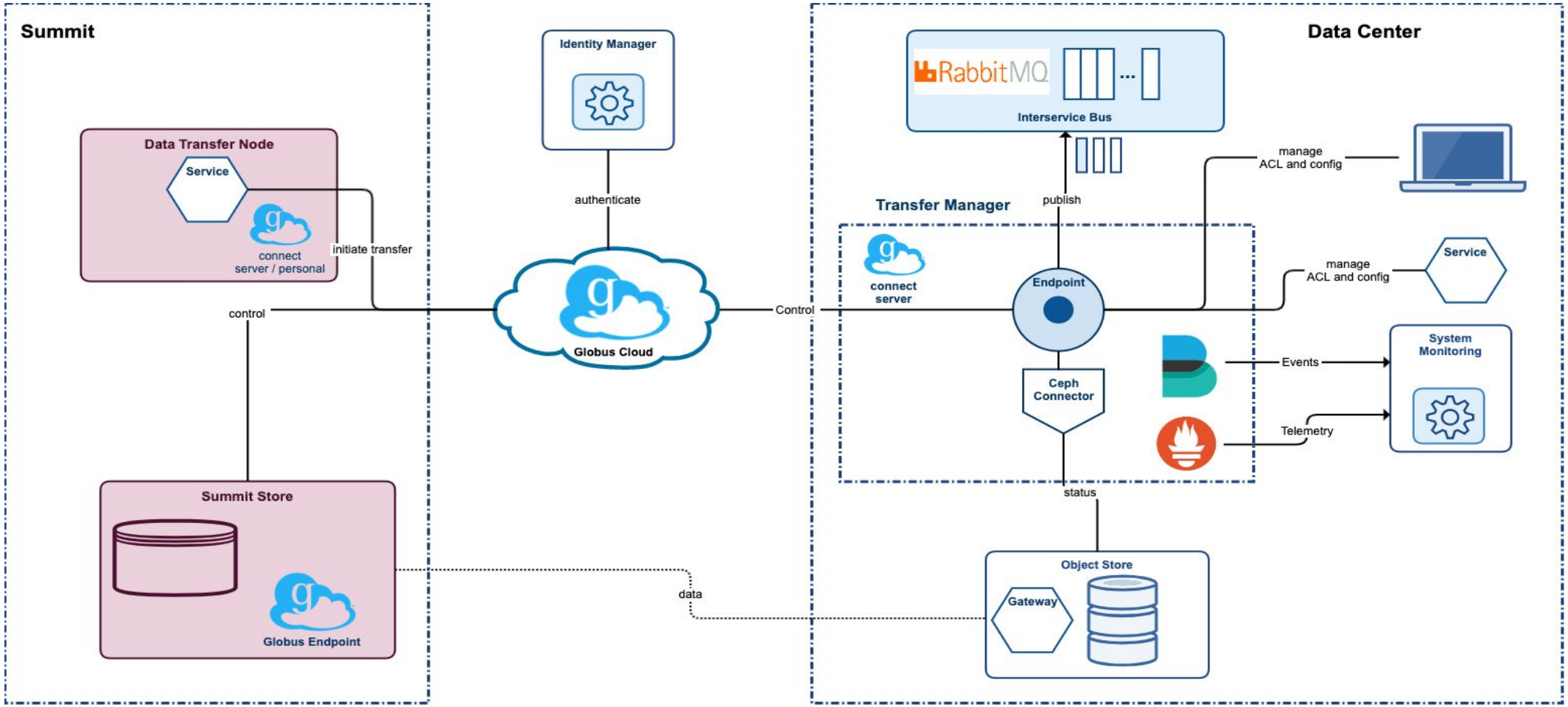
Summit Data Transfer Process



Goals of DKIST DC Ingest Process

- Receive data sent from the Summit to the Data Center
- Verify that data sent is data received and notify summit of result
- Route Science Data
- Route Transfer Manifest data
- Ingest Science Data
- Ingest Transfer Manifest Data
- Facilitate validation of stored data integrity
- Facilitate management of failed ingest(s)
- Make valid science data received at the DC searchable by a DKIST Authorized Agent within 10 days

Summit to Boulder Transfer



Data Transfer from Summit to DC

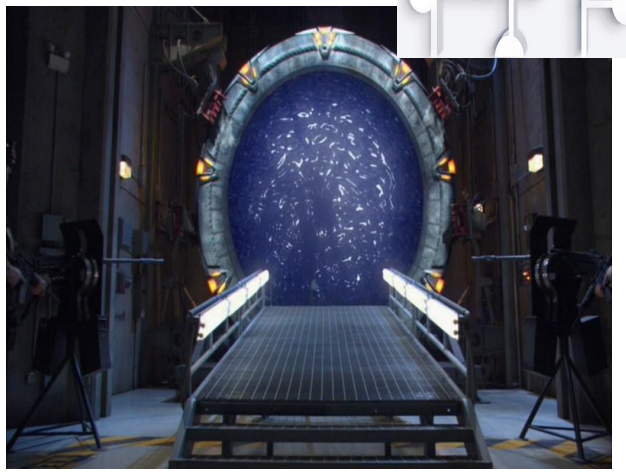
- Data Transferred for summit is packaged in accordance with an ICD

  /YYYYMMDDHHMM (Datetime at start of transfer)
  /Control (contains manifest of files to be transferred)
  /Transfer (contains FITS files)
  /Camera (contain specially collected VBI FITS files)
  /Archive (contains tar balls)

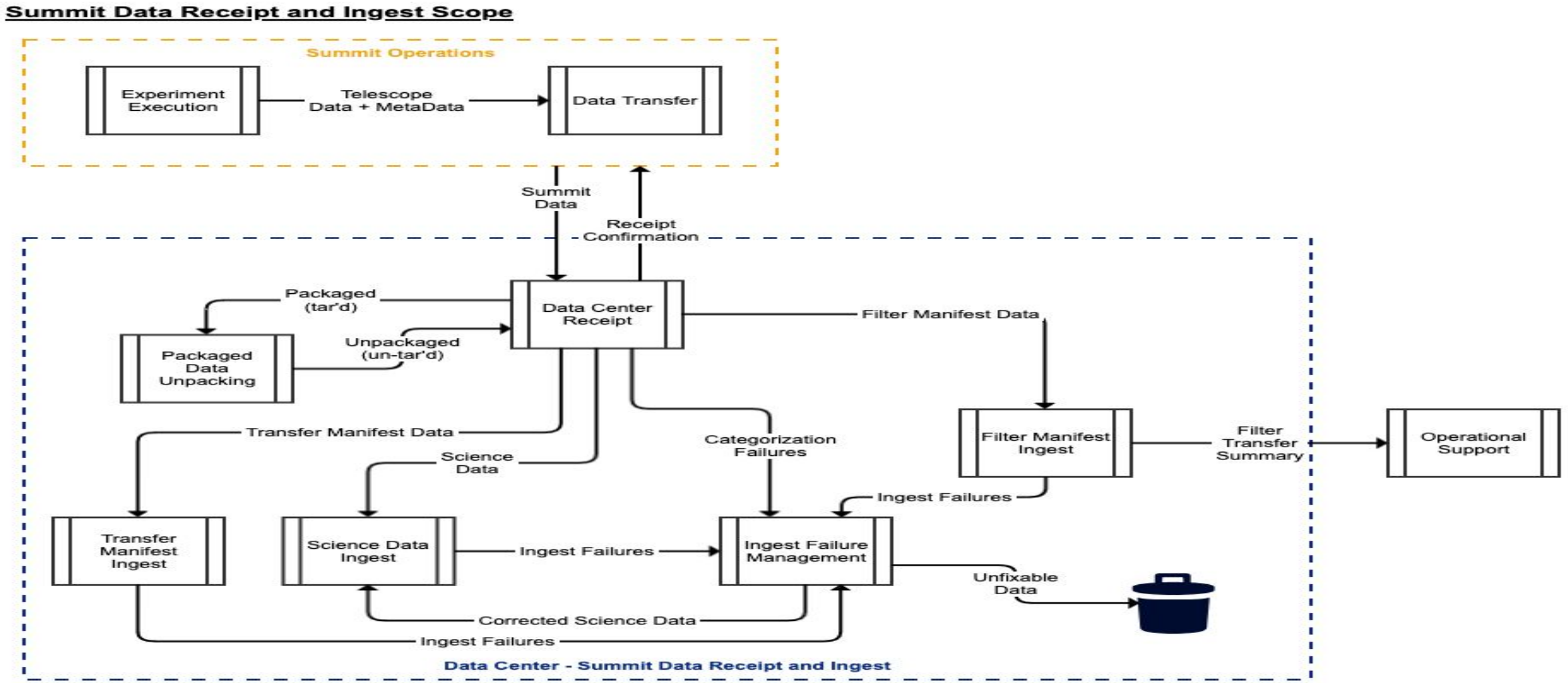
  - Each FITS file is uniquely named as *$Transfer/$ExperimentID/$ObsProgramID/$InstrumentProgramID/$FrameID.fits*
- Data format is FITS - DKIST Spec (122)
- Received files are directed to the “inbox” bucket in the DC Object store

Data Lifecycle

- Raw data produced by telescope
  - Transferred to Data Center (DC)
  - Deleted
- L0 data ingested by DC
  - Stored permanently in immutable object storage
  - Backed up to AWS Glacier
- L1 data created by DC
  - Stored as mutable data
  - Not backed-up
  - May have different versions due to reprocessing
  - Version N stored and visible through data portal to PIs and public
  - Version N -1 stored but not visible
  - Versions N - 2 or earlier inventory records kept – tagged as “deleted”
    - Science data deleted



Data Ingest Process



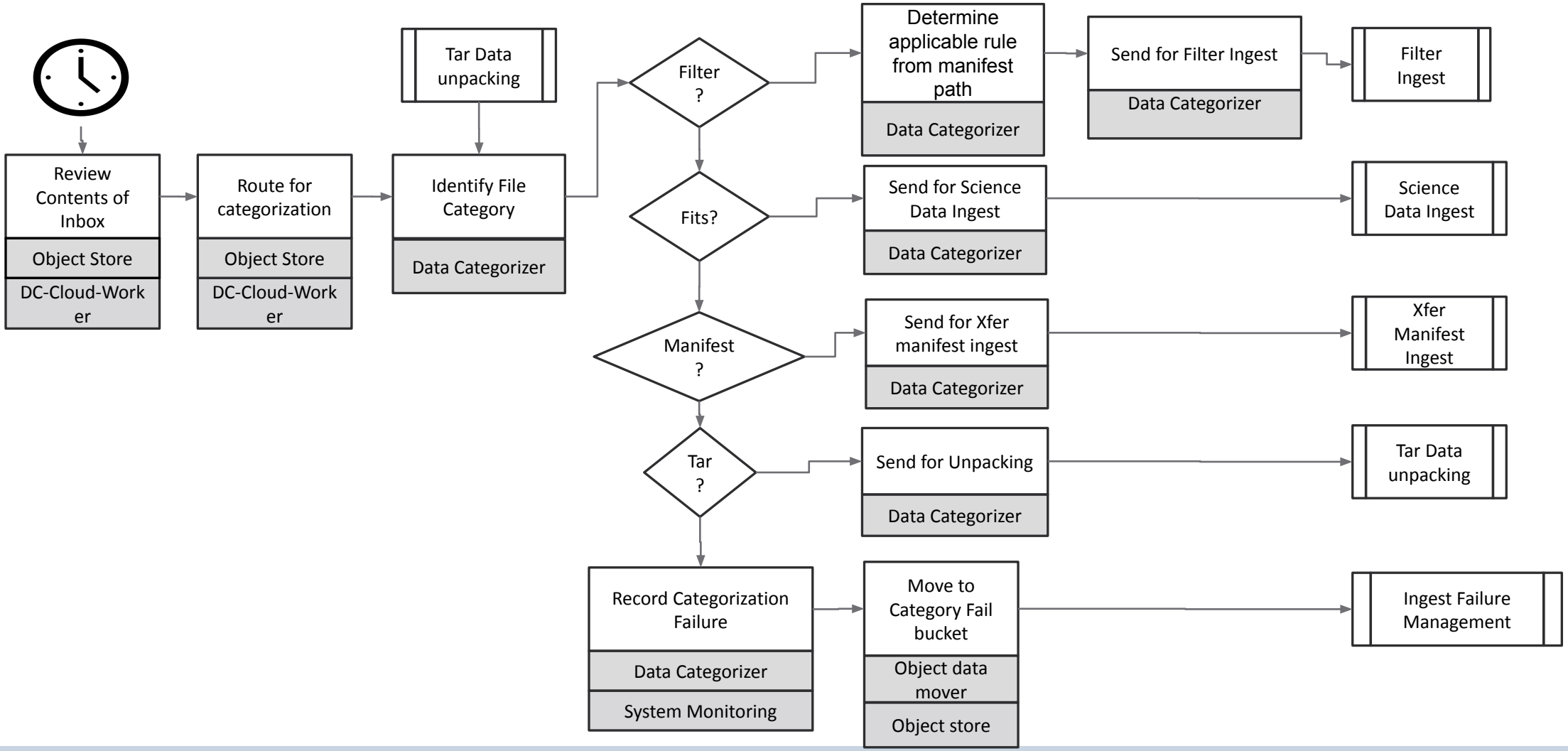
Ingest Process Scope

- Verifiably receiving data from the summit
- Categorizing the data received
  - The three defined types of data that are being received from the summit
    - Science Data
    - Transfer Manifests
    - Tar'ed files
- Each type is then routed to its appropriate ingest pipeline and stored in the appropriate data store.
- In case of Failure – routed to Failure buckets for human intervention
  - schema (spec 122) validation failure
  - file category unrecognized

Getting Data to the DC Inbox

- Transfer is initiated by the Summit personnel
- Transfer is handled by Globus
  - Globus manages authorization/authentication
    - DC holds no PII – one less security issue to worry about
  - Globus Manages the entirety of the transfer
    - Size of transfer is immaterial
    - Completion of the transfer (including recovery from interruptions)
    - Verification of Data Transfer (number of bytes, checksums, etc.)
    - Notification of Xfer success/failure
  - Transfers parallelized to maximize bandwidth utilization
- Once the transfer is complete and verified (or not)
  - Success (or failure) message is generated
    - If failure, any partial transmissions are deleted

Inbox Data Management



Filtered Data

- The ingest process begins by filtering out data that should not be “ingested” into the data center and its inventory
- Filtering was not part of the original design
- Issues with VBI data necessitated this “bolt on”
  - Issue is VBI camera noise
    - Upshot is that VBI data could not be processed (to L1) in real time on the summit
- VBI processed (L1) data is 1/80th the size of raw VBI data
- Raw data currently diverted to the DC for speckle reconstruction - outside of the data center
- DC was getting data it was never supposed (or designed) to have – and having to make it available to an external entity for processing - with the follow-on requirement to re-ingest processed data from a non-summit location
  - Issues with manifests and counts
  - Issues with routing
  - Issues with re-ingestion and possible deletion of raw data (expressly forbidden)
  - Issues with possible “re-re-ingestion” of reprocessed VBI data (never a possibility with real time reconstruction)

Transfer Manifest Ingest

- The Transfer Manifest Ingest process is responsible for
  - storing the expected frame counts by observingProgramExecutionId (OPXID).
  - Ingest errors arise from schema validation issues
- The manifest is used strictly for counts
- Files received are not matched 1-1 against those contained in the manifest

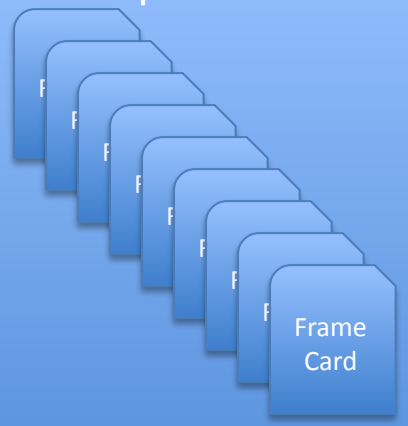
Science Data Ingest

- The Science Data Ingest process is responsible for
  - Verifying FITS headers meet spec
  - ingesting unfiltered Science Data
  - inventorying Science Data storage location.
- Science Data headers are validated against a schema as defined by Spec-122
  - Frame headers are appended with select process management metadata to aid in inventory regeneration (if necessary).
- The frame, with its appended headers, is then stored in the object store under a "folder" named after its OPXID
- An Inventory record is created which includes
  - Frame headers
  - location of the file
  - date
- Lastly a counter is incremented to track how many files have been received (by OPXID) to be counted against manifest

L1 Processing Management

ViSP recipe

Input Dataset



ViSP parameters

General DKIST parameters

DC CDR

TOOL: WORKFLOW MANAGEMENT

Apache Airflow

ViSP recipe

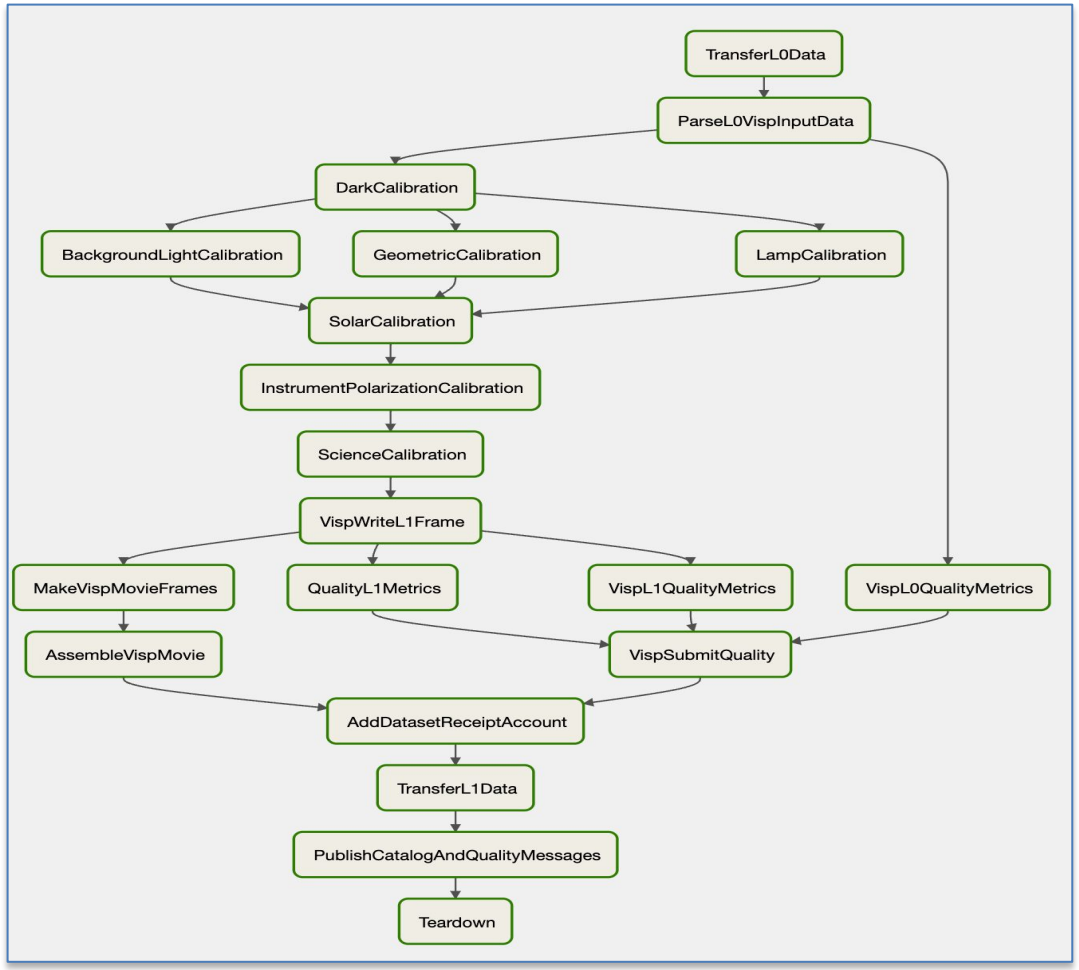
VBI recipe

VTF recipe

L1 Recipe Run

Apache Airflow

ViSP recipe



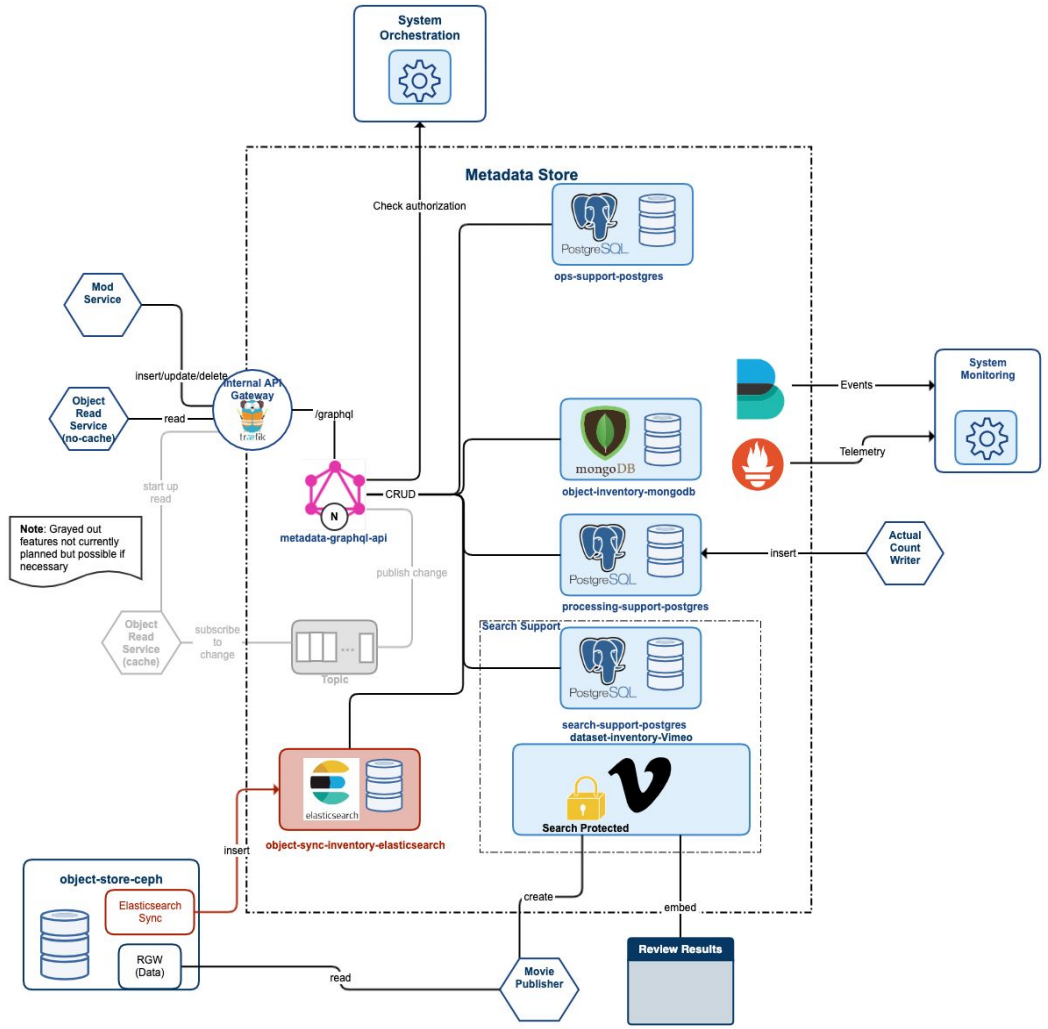
Data Inventory

DKIST Data Inventory Definition

- The DKIST data inventory is a detailed catalog of all the science data assets Currently held by DKIST DC, designed to help Data Center personnel, scientists, and users quickly find the most appropriate data for their purpose.

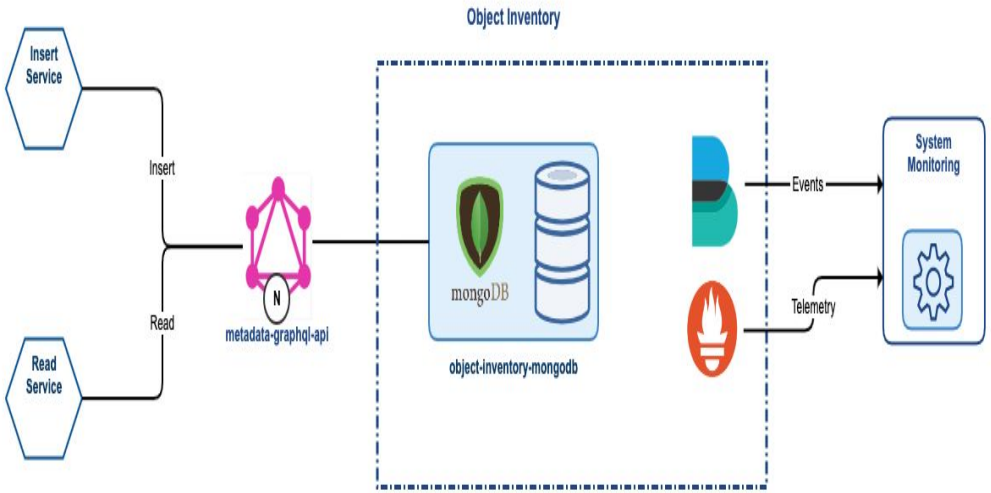
Data Storage Structures

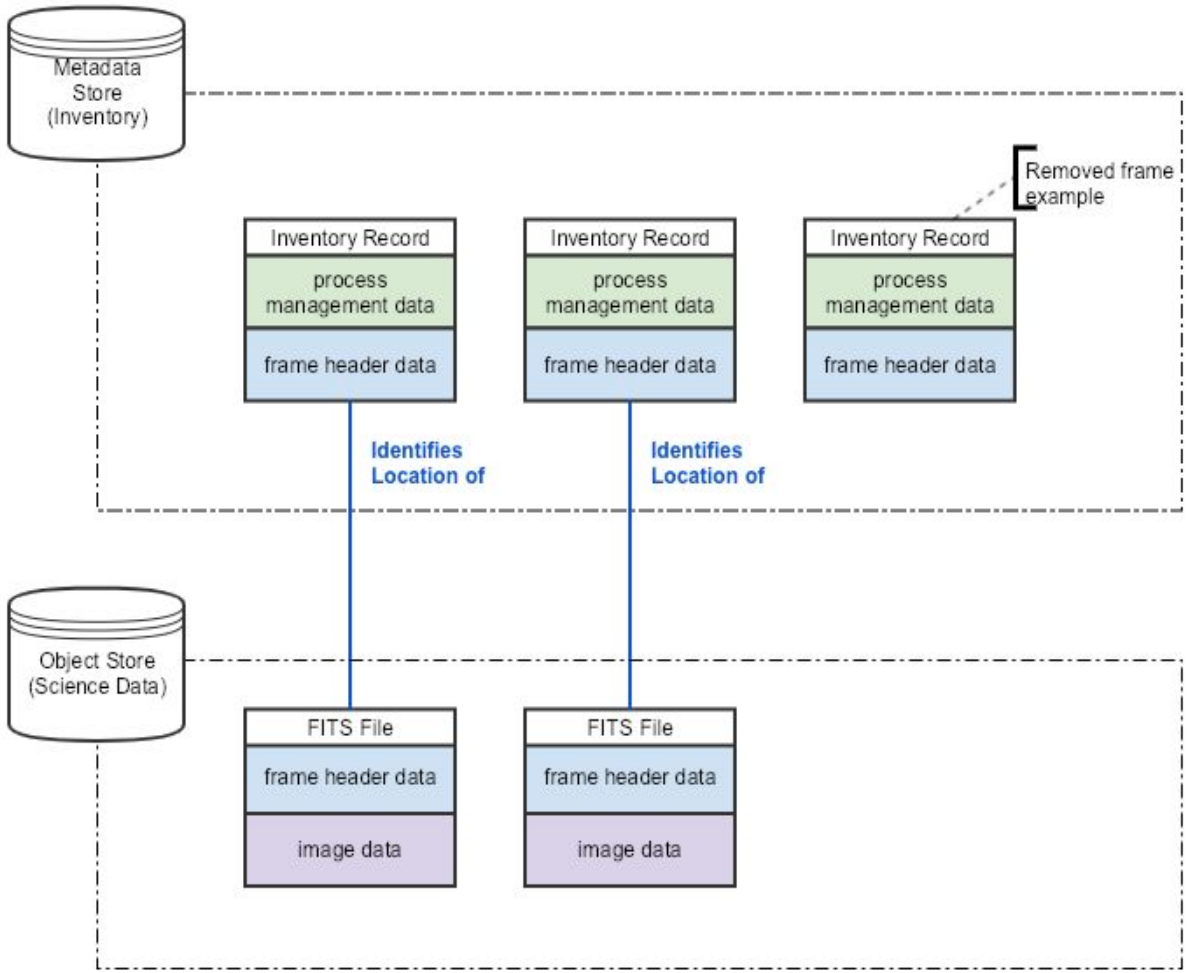
- Object Inventory
  - catalogs all objects in the Ceph cluster
  - Implemented on Mongo DB (collections)
    - Frame Inventory
      - Stores a copy of FITS headers
        - JSON documents – indexed for easier search
      - Stores locations (paths) of frames
    - Object Inventory
      - all other objects that are not FITS frames
- Search Support DB
  - Implemented in Postgres
    - Stores all data required to perform search and discovery of L1 data
- Support DB
  - Implemented in Postgres
    - Stores all metadata required to allow processing of L0 data
    - Stores all metadata related to the creation of L1 (provenance)



Object Inventory

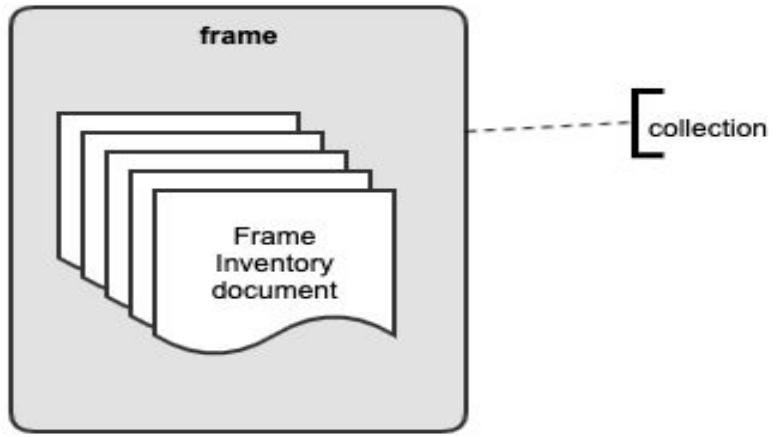
- Science Data is big both in volume (PB) and count (millions of files).
- Data frames are stored in Ceph object store
- Location record are placed in inventory for later retrieval.
- The inventory component enables discovery and select metadata analysis

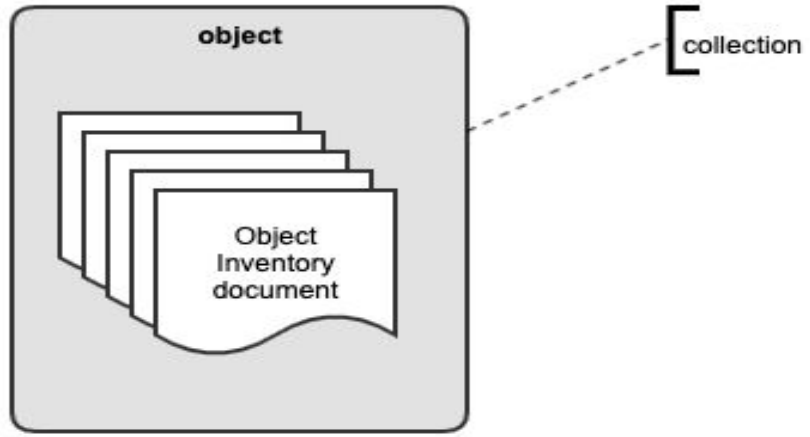




Object Inventory

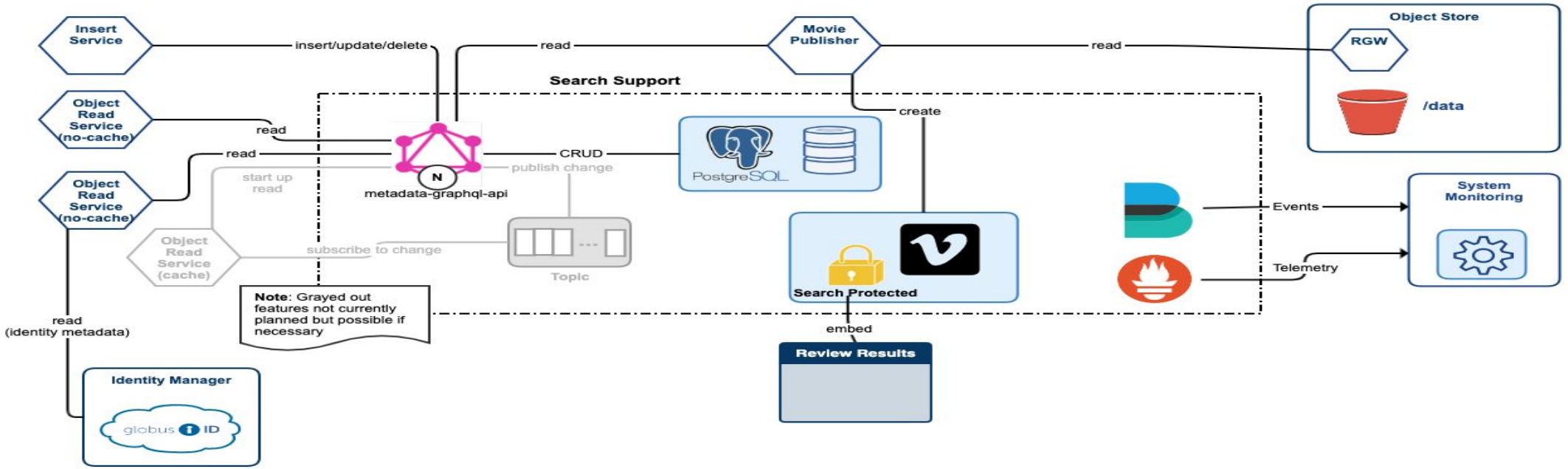
- The Frame Inventory data store consists of a document database supported by MongoDB
- Mongodb data is allocated as documents. The following entities are modeled as documents:
  - Frame Documents –
    - One per FITS file in the object store.
    - Contains a copy of all FITS headers
  - Generic Object Documents
    - Advanced Scientific Data Format (ASDF) Documents - 1 per ASDF file in the object store. Only contains pointer information
    - Movie - 1 per browse movie in the object store. Only contains pointer information
    - Quality report
    - Large parameter sets for processing





Search Support

- The Search Support data store consists of a relational (Postgres) database
  - The data model for the RDB contains data associated with the discovery of data, its quality, and related embargo enforcement.
  - Part of the discovery process is the presentation of Browse Movies hosted on Vimeo.
  - DB queries and results facilitated by a RESTful API sitting on top of the metadata-store-api (graphql)

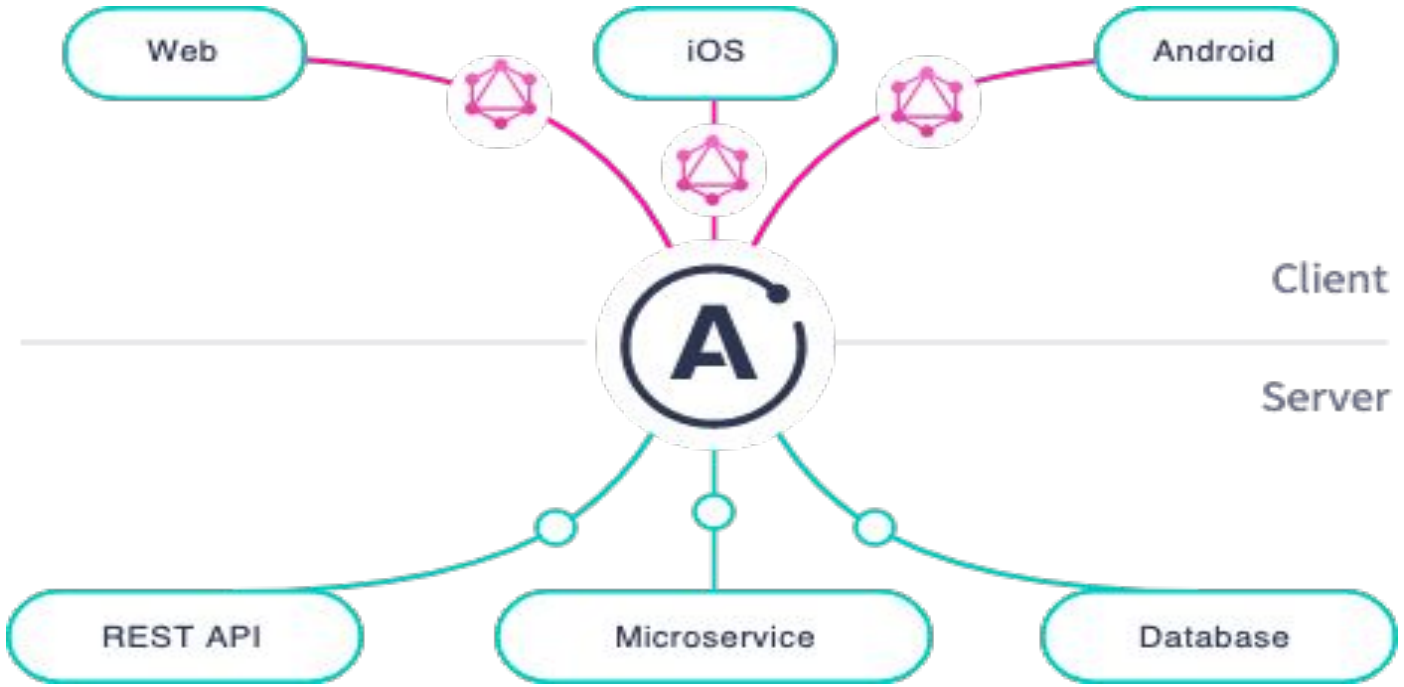


The Search Support DB

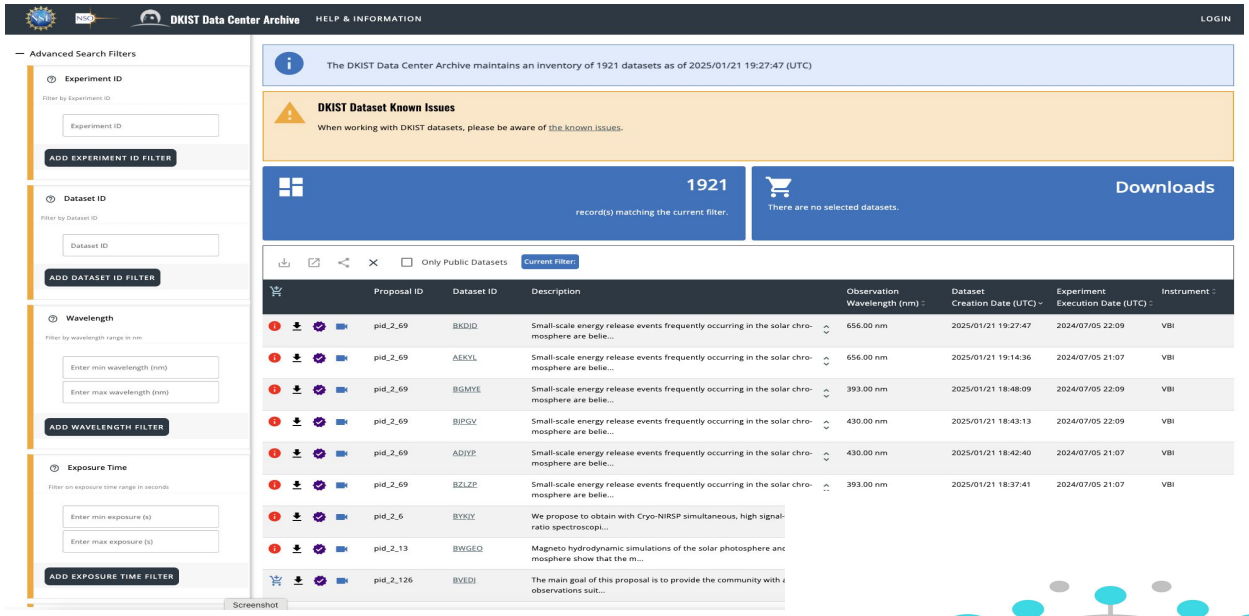
- Places metadata about frames/datasets in a searchable database
- Allows the creation/addition of of data (L1) and metadata including from from other sources (e.g. Ops tools)
  - Observation descriptions
  - Who is allowed access and for how long
  - Dataset provenance
    - Pipeline version
    - Date of creation
    - Original L0 Data
    - etc
- Allows internal and external entities to perform a search of DKIST DC data holdings (via API)
- Allows the DC to perform periodic audits of the data stores to assess possible losses
- Allow internal and external entities (e.g. User Tools) to download metadata (FITS headers) about data products without having to download frames

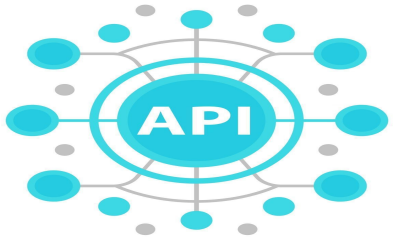
GraphQL Framework

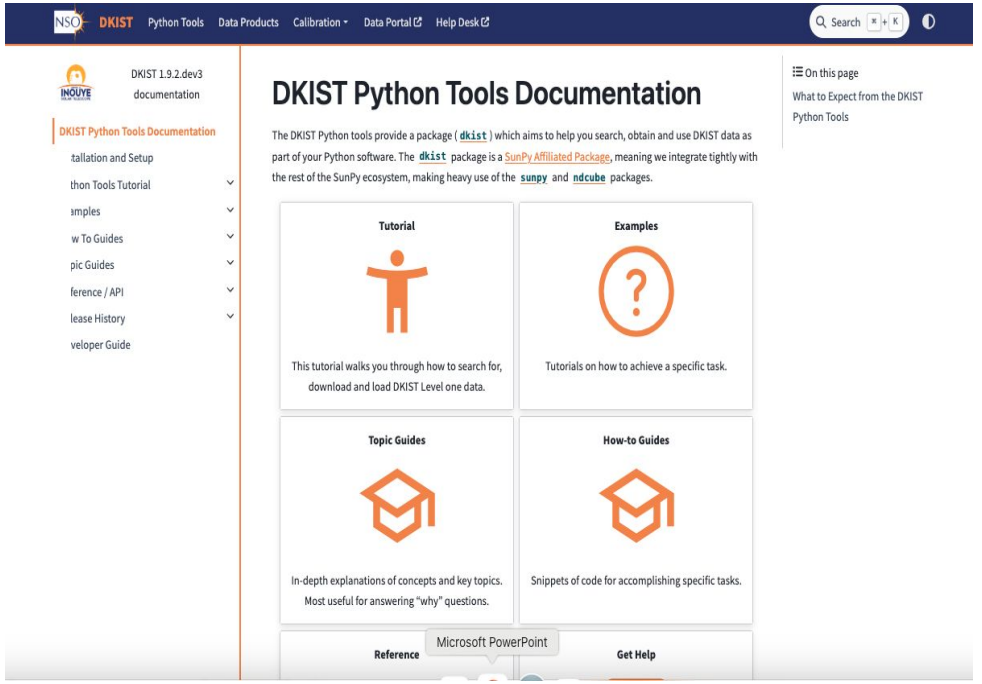
- Works around REST downsides
  - under/over-fetching
  - multiple requests for multiple resources
- Abstracts the data models / management systems from the client code



Distribution





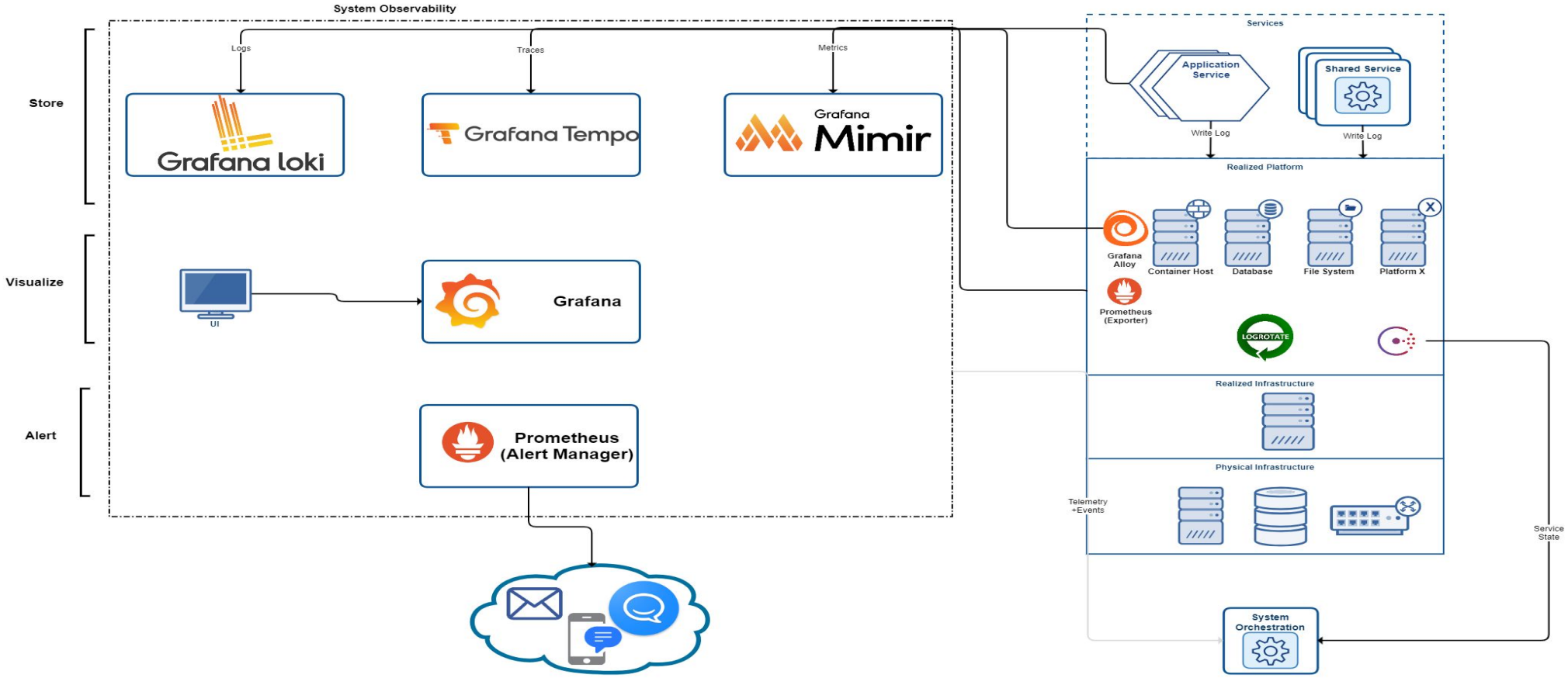


DC Observability

What is observability ?

Observability is the ability to understand the internal state of a system based on its external outputs, such as logs, metrics, and traces. It involves collecting, analyzing, and visualizing data to gain insights into system performance, behavior, and security, enabling effective troubleshooting, anomaly detection, and optimization.

Observability Infrastructure



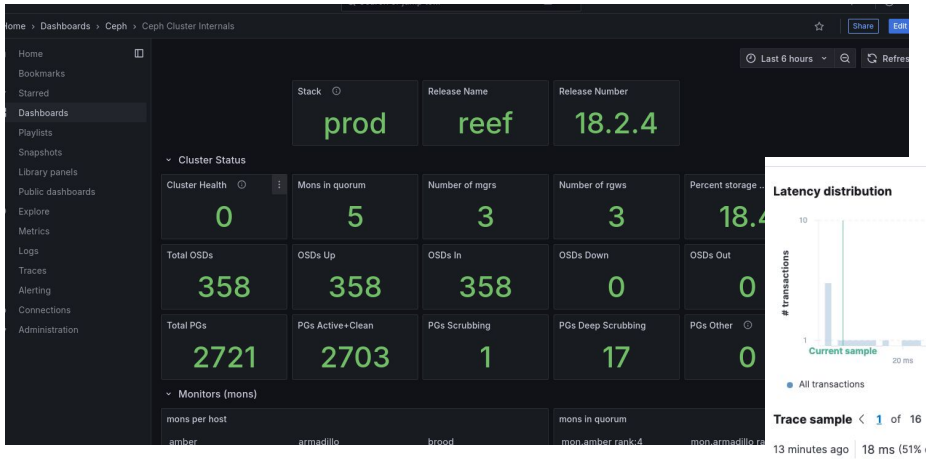
Overview of DC observability

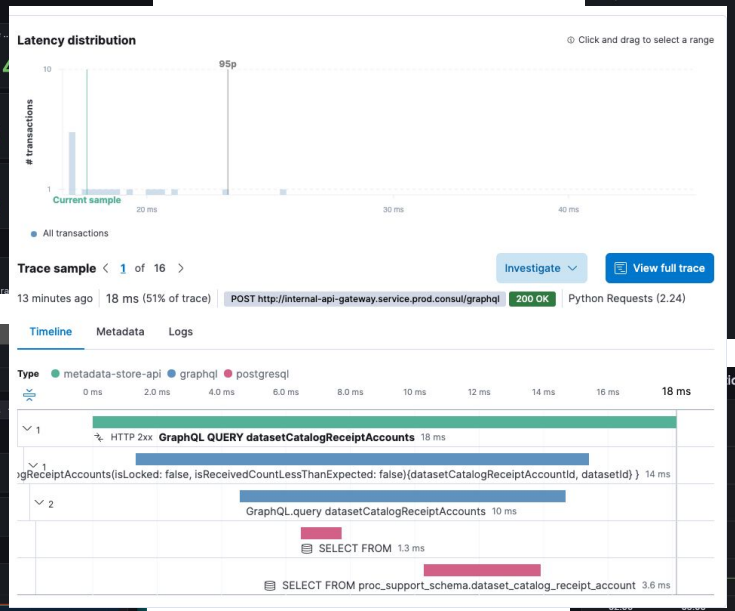
- Logs
  - Structured or unstructured text records of events
  - Can be out-of the-box application logging or purpose built
- Metrics/Telemetry
  - Values represented as counts or measures
  - Measures core system functions (eg % CPU, memory used, etc.)
- Distributed Tracing
  - Instrumented code that exposes activity of a transaction or request as it flows through the system.
  - Highlights how services perform.

Monitoring and metrics

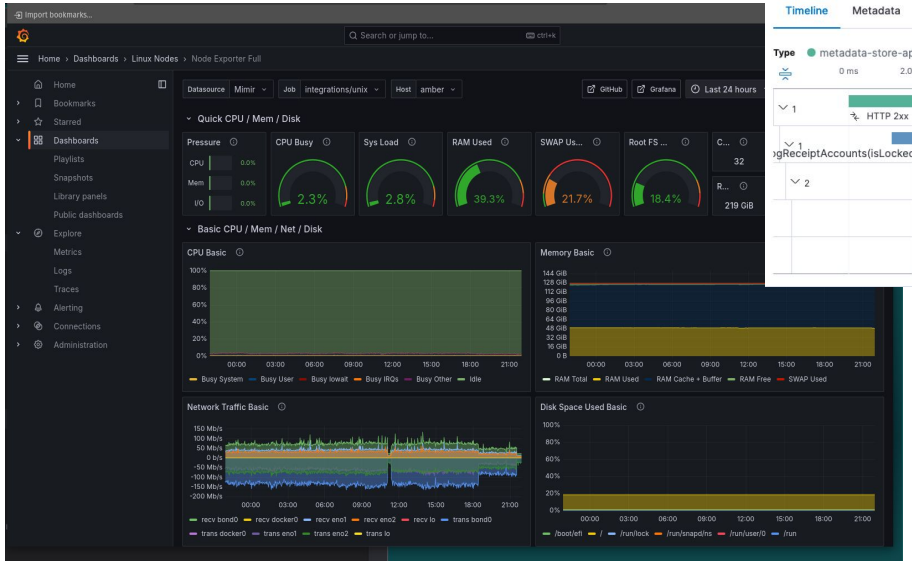
- Metrics collectors as close to the generation as possible.
  - Infrastructure Metrics
    - Host (CPU, Network, Memory)
    - Ceph Object Store
    - Gluster File Store
    - RabbitMQ Event Bus
  - Application Metrics
    - Event Telemeter
      - Generates metrics based on events flowing through event bus
    - Metadata Telemeter
      - Generating metrics based on queries to larger system or aggregated pieces of infrastructure

Dashboards


Home › Dashboards › Ceph › Ceph Cluster Internals
Stack
prod
Release Name
reef
Release Number
18.2.4
Cluster Status
Cluster Health
0
Mons in quorum
5
Number of mgrs
3
Number of rgws
3
Total OSDs
358
OSDs Up
358
OSDs In
358
OSDs Down
0
OSDs Out
0
Total PGs
2721
PGs Active+Clean
2703
PGs Scrubbing
1
PGs Deep Scrubbing
17
PGs Other
0
Monitors (mons)
mons per host
mons in quorum


Latency distribution
Click and drag to select a range
95p
transactions
Current sample
All transactions
Trace sample 1 of 16
Investigate
View full trace
13 minutes ago
18 ms (51% of trace)
POST http://internal-api-gateway.service.prod.consul/graphql
200 OK
Python Requests (2.24)
Timeline
Metadata
Logs
Type
metadata-store-api
graphql
postgresql
HTTP 2xx GraphQL QUERY datasetCatalogReceiptAccounts 18 ms
GraphQL.query datasetCatalogReceiptAccounts 10 ms
SELECT FROM 1.3 ms
SELECT FROM proc_support_schema.dataset_catalog_receipt_account 3.6 ms


Home › Dashboards › Nomad › Nomad Allocations
CPU
CPU %
CPU Usage (Cores)
CPU Throttled Time
RabbitMQ - 288b5941-e75b-b0a0-5159-b2362f399043
RabbitMQ - 36c3e5a7-b66e-96d4-9a68-dc136d60daf9
RabbitMQ - 49910fc8-3b90-7e34-d3d8-f36f7d2f6f2a


Home › Dashboards › Linux Nodes › Node Exporter Full
Quick CPU / Mem / Disk
Basic CPU / Mem / Net / Disk
CPU Basic
Memory Basic
Network Traffic Basic
Disk Space Used Basic


Science Frames ready for ingest
Science Frame Ingest
Success
Failure
Frames Archived
Total

Current and Future Challenges

- Current Challenges
  - Small team juggling Ops and Dev
    - 4 FTEs working Systems and DC Infrastructure
  - Unforeseen changes (eg-VBI processing off summit)
  - Multiple modes of operations for each instrument
    - Often causes L1 processing issues – and delays data publication
- Future Challenges
  - Higher data rates as instruments are upgraded
  - Limited footprint – no physical growth possibility