iRODS at Bristol Myers Squibb

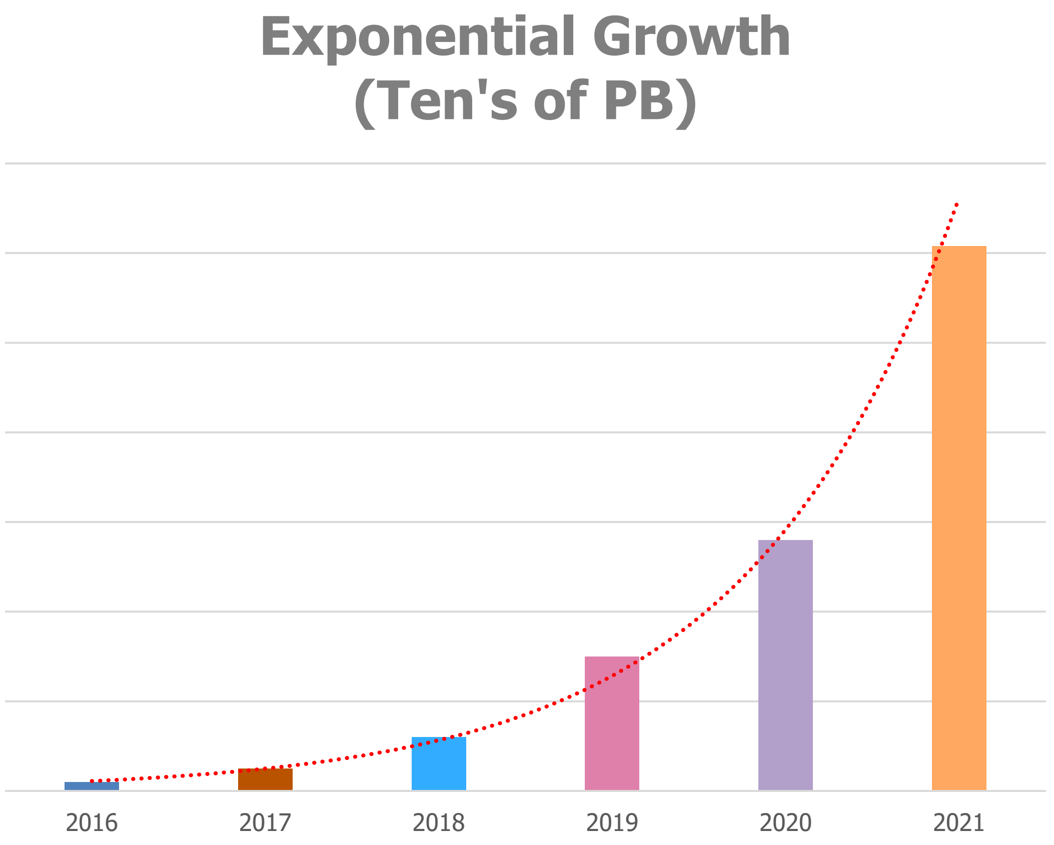
Overview of Projects. Leveraging iRODS for Scientific Applications in Amazon Cloud

Radha Konduri | Dmitry Khavich

Data Stores & DataOps, R&ED Data Management Platforms

iRODS UGM, Jun 8-10, 2021

It's all about data, Big Data!



From GB's to PB's scale

Data governance

- 25+ years of retention with Backups

Scientific data sets

- NGS data
- Proteomics
- Flow Cytometry
- Imaging data
- High-Throughput screening
- Mass spectrometry
- Databases

Major data sources

- Raw data from labs
- Scratch space
- Results data
- External collaborations
- Public & government agencies
- R&D

Lab data challenges

Data insights are only as good as the data that drives them

Data accessibility and sharing

- Silos between teams (organizational resistance)
- Image acquisition systems are local, and self contained

Computing power, Networking & storage

- Efficient data exchanges, storage and processing

Replicating results

- Reliability, efficiency, speed, provenance & repeatability for testing & validating

Data mining

- Lack of good metadata annotation
- Image datasets are hard to find, difficult to extract

Data standards & compliancy

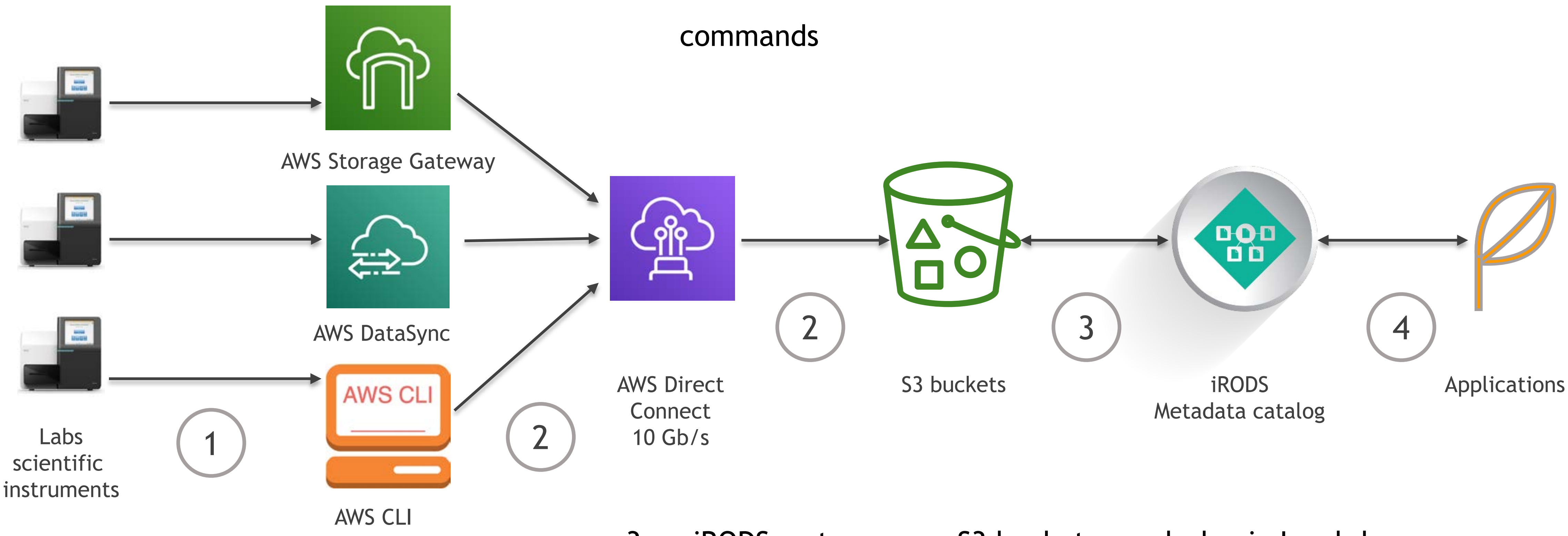
- Different formats, data integration and validation

Data Management

- Searching, grouping, updating metadata via UI application

Typical data flow diagram

1. Instruments writes raw data into local scratch space
2. Raw data pushed to S3 by Storage Gateway/DataSync or via AWS CLI S3 commands



3. iRODS system scans S3 buckets regularly via Lambda
4. Applications request data via iRODS metadata catalog

Use case: Lab Data Management for Immuno-Oncology

Immuno-Oncology teams require :

- Diverse, data-rich discovery research in CAR/eTCR discovery, T cell engineering, and preclinical studies.
- Tools and data standards to effectively aggregate data across various assays, track data lineage, and automate the acquisition and movement of instrument data.

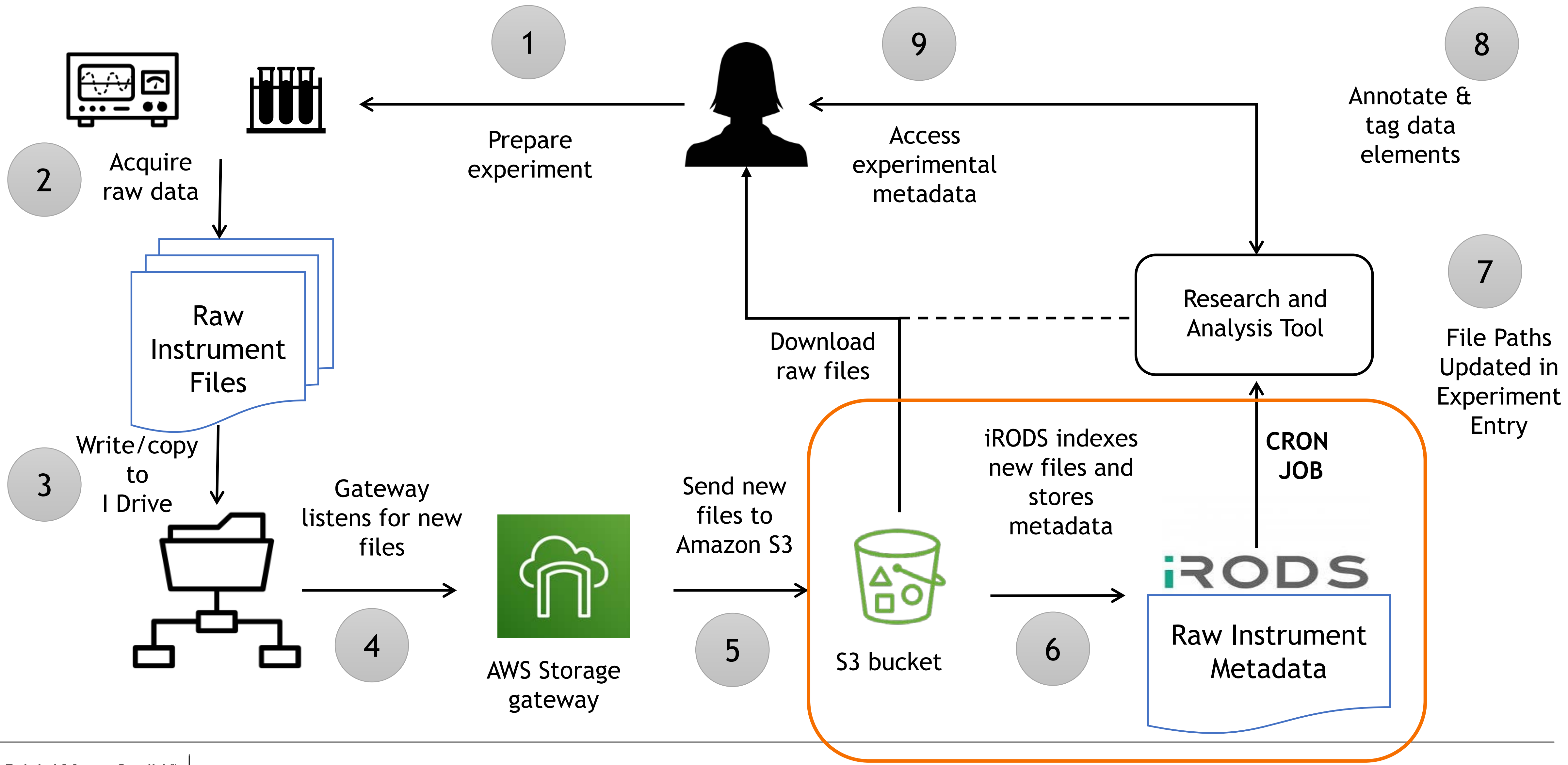
The capability has three major components:

1) Lab Data Hub, a platform for archiving, and curating raw data;
2) Data Model, a model to guide the entry and semantic harmonization of lab data; and
3) Discovery Work-Bench, an analysis tool for aggregating data across experiments.

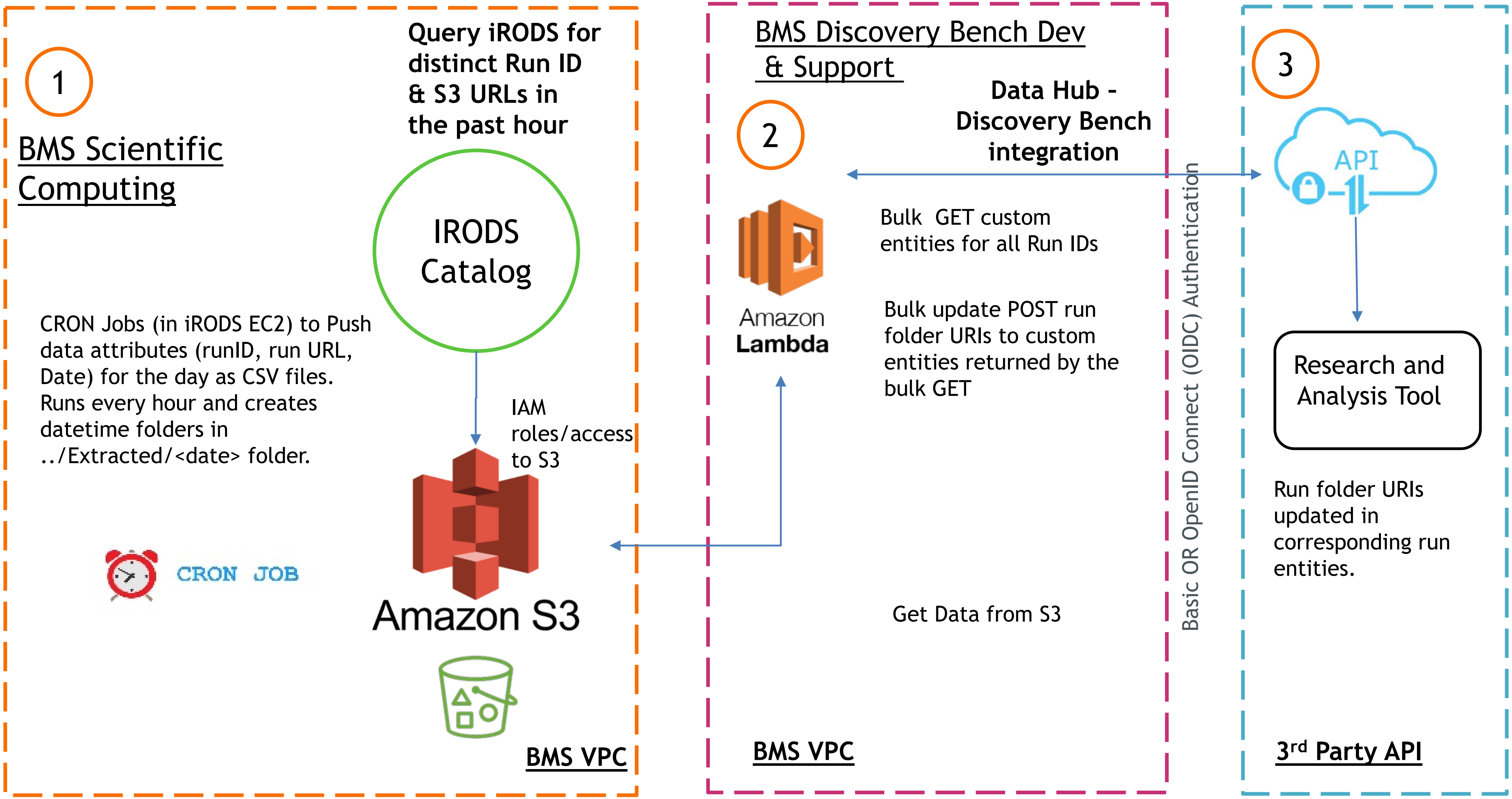
Measures of Success:

Quantitative: Productivity value metrics
Qualitative : Improved decision-making, and risk compliance

Business Process & Solution: Lab Data Hub Architecture



Business Process : Data Flow



Metadata Tagging Requirements

Different ways iRODS extracts the metadata

- Metadata values are extracted from file path in the bucket and file name using regular expression.
- Time of creation/modification and size of the file is parsed from AWS S3 head object response.
- A unique tag for each file is generated using the python hash library for the contents of the file.
- Files like XML, JSON, CSV and TXT are read for setting the metadata for the data objects or collections.

Process of CSV file generation for analysis tool integration

iquery runs every hour against iRODS collection.
Obtains distinct combinations of metadata like experiment run ids and run folders that fit the criteria.
The CSV file is loaded in the S3 bucket location.
The research and analysis tool is configured to read the latest CSV file from S3 bucket.
Scientists are able to analyze the data and generate the reports.

Challenges and solutions:

SQS issues:

- Unable to purge stuck messages
- Single SQS queue had messages from all projects
- Purging the queue would purge legit messages of other projects and affect them
- Solution: Separation of SQS queues only by project but also by environment like UAT/prod

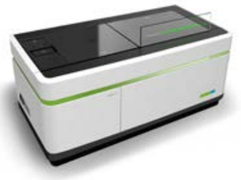
Issues generating CVS file:

- Special char in files names – like quotes, percent, parenthesis, spaces and especially commas
- Limitations of iquest arguments: revolved by counting the number of files before creating the argument and grouping arguments by 3000

Performance/tagging issues:

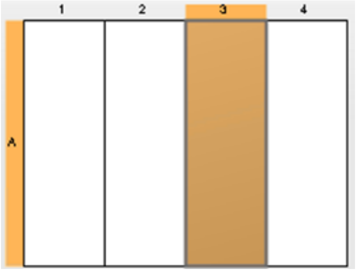
- Improved the throughput of the iRODS delay queue by using a larger instance type and
- Modified the parameter 'maximum_number_of_concurrent_rule_engine_server_processes' accordingly.
- Python code modification for tagging empty files.

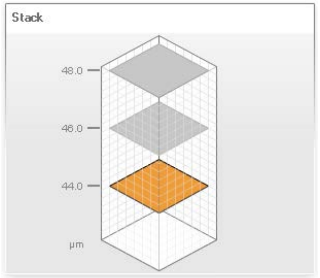
Use case: Enabling LDO Tissue Imaging and AI



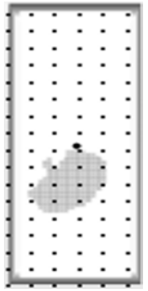
On Instrument Computer
- Scientific Database
- Scientific Software

Insert “Plate” with 4 slides, designated as wells A1, A2, A3, and A4

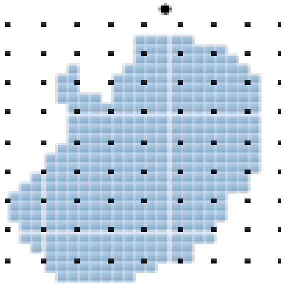




For a specific slide (well), low res scan to locate tissue sample.
PreciScan

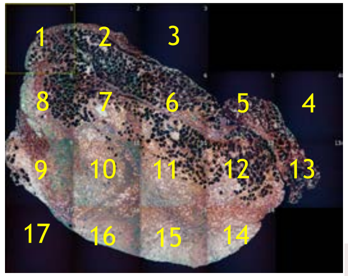


High resolution imaging of tissue sample



Each tiny blue square is a field. Each field will have a different image for each color (channel) and for each confocal z-plane

Building Blocks software “stitches” fields together



iRODS

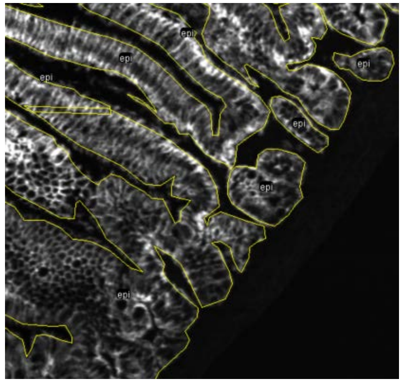
Search UI

Data Analysis

python

R

UNet Machine Learning



Generic image of a tissue

Approach to enabling the next generation of data management

Medical scanner, microscopes

High content Screening instruments

iRODS

S3

Implement the integrated COTS Rule Oriented Data System (iRODS) with petabyte-scale capabilities to automatically ingest imaging and analysis data sets directly from the instruments and analysis tools

AIMS

RunDef

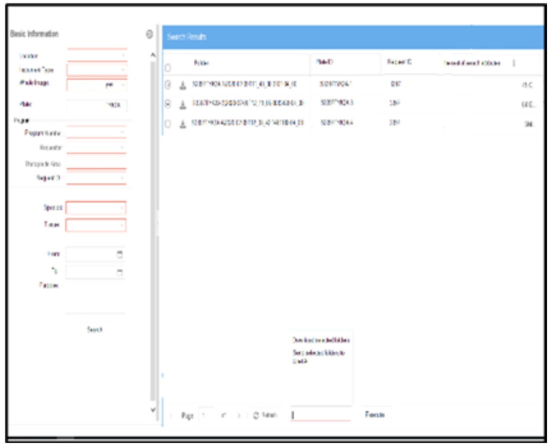
iRODS

S3

SARP

ARIA Study

Enable iRODS to automatically ingest, validate, and assign metadata to image datasets and provide provenance



iRODS

S3

On top of iRODS, build user friendly interface to provide intelligent search and enable user to request variety of operations on images

Image Transformations

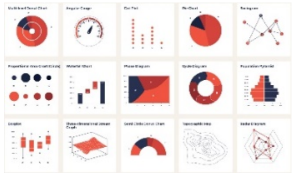
Electronic Lab Notebook ELNB

HALO

iRODS

S3

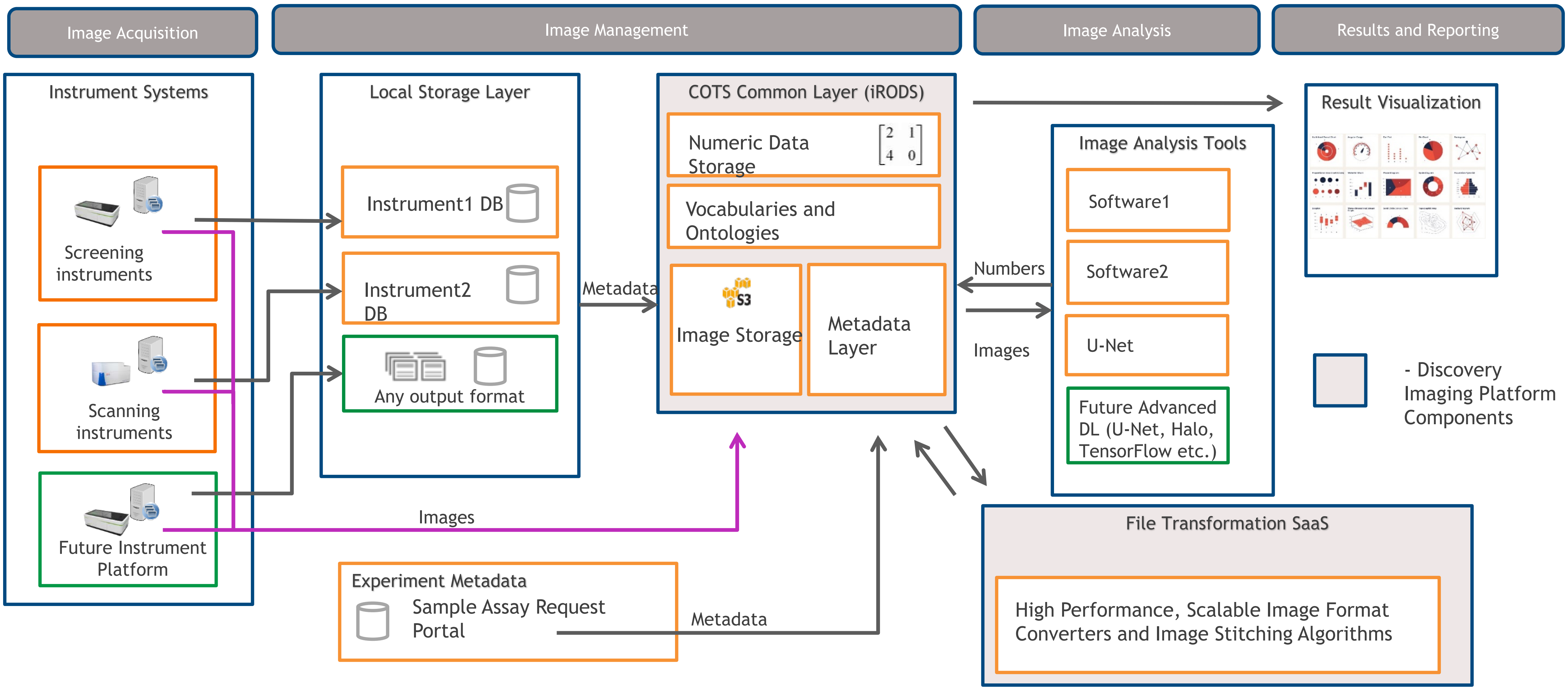
TensorFlow



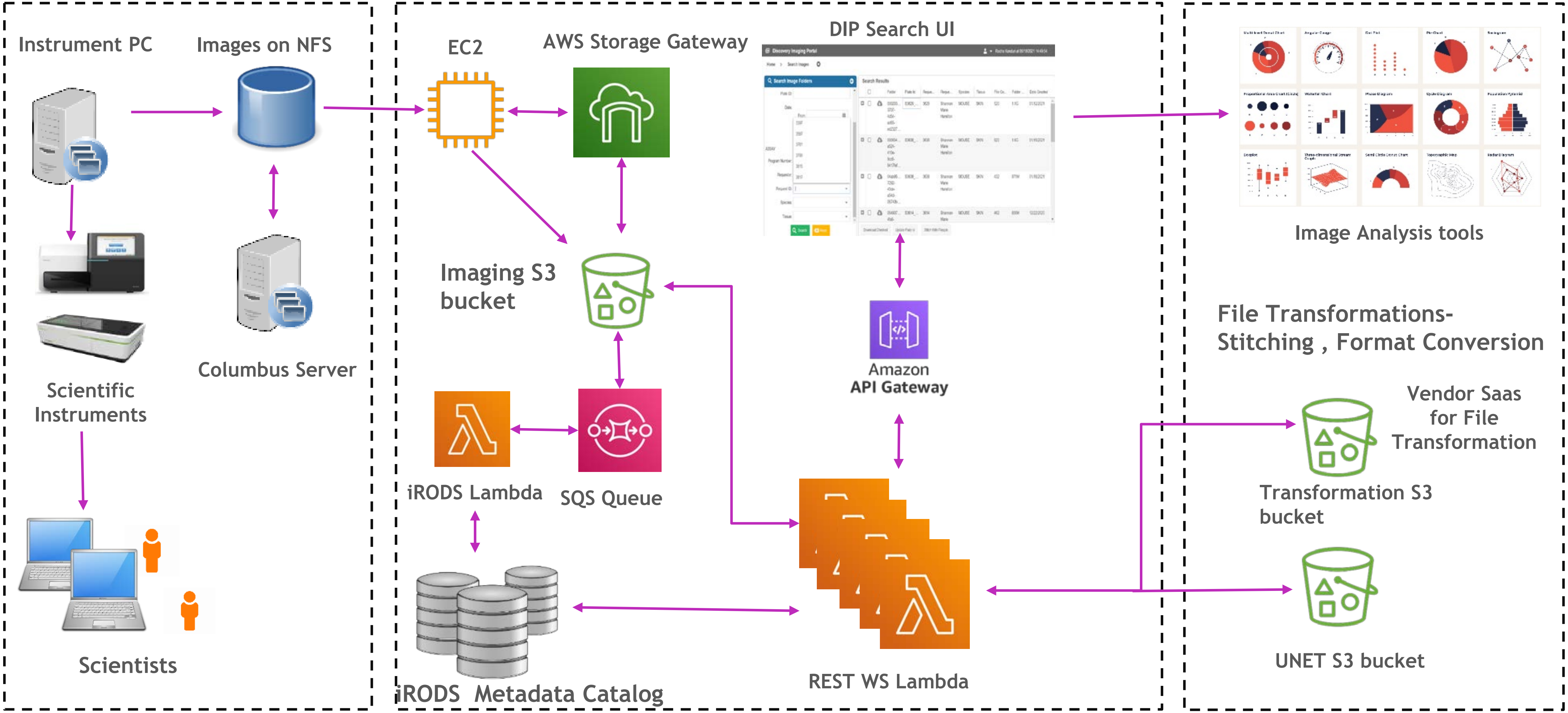
Data Visualization tools

Deploy and integrate with iRODS easy configurable services to enable data analysis pipelines, perform image format conversions, create image montages, and other image manipulation as required

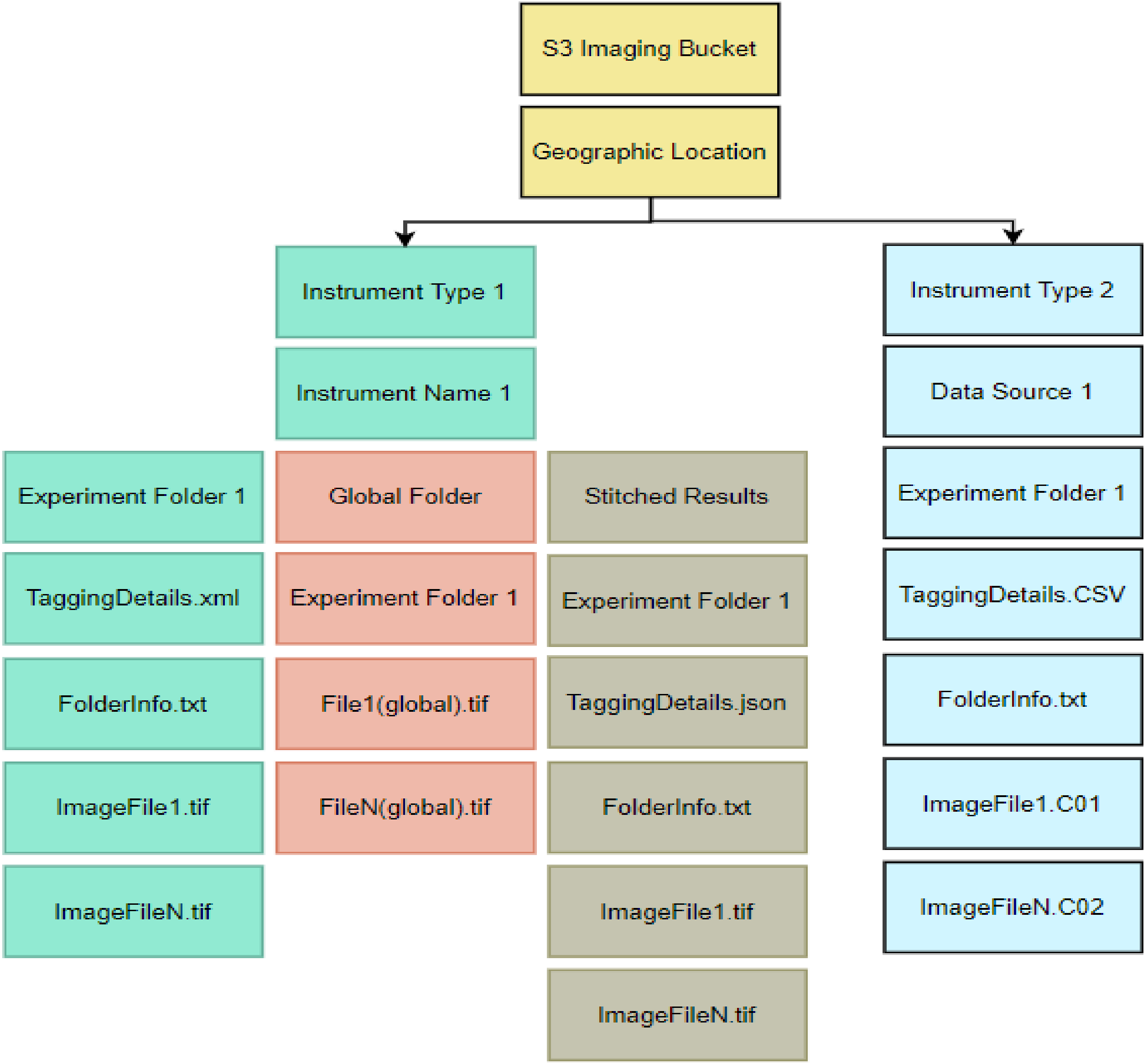
High level Architecture



High level flow



Model overview for File tagging



4 Data types :

- Instrument1 Original
- Instrument1 Stitched
- Stitched Results
- Instrument2 Original

Tagging Process :

- The dynamic Policy Enforcement Point (PEP) defined in the loaded rulebase file core.py : pep_api_phy_path_reg_post
- The rules logic is defined in the PEP function.
- A file gets registered in iRODS
- The core.py triggers and the PEP adds the task to the delay queue.
- The tasks in the delay queue execute.
- Depending on the type of TaggingDetails file, the data type is determined.
- The tagging of files/data objects and folders/collections occurs.

Challenges , solutions & performance optimizations:

Performance issue when searching for files with a specific Attribute values

- Changes in the data model
- Set common tags at the folder/Collection level for easily finding the dataset of interest
- Set file specific tags to the files/DataObject within the Collection.
- The search portal has filters that can be used to search for the specific folders
- Optimized query to search for specific attributes like DataObjectMeta.name rather than DataObjectMeta.

File	Folder	Attribute
yes		FILE_NAME
yes		CHANNEL_NAME
yes		WELL
yes		FILE_EXTENSION
yes	yes	SPECIES
yes	yes	PROGRAM_NO
yes	yes	SAMPLE_TYPE
yes	yes	TISSUE_TYPE
yes	yes	THERAPEUTIC_AREA

Performance Metrics *:

Before:

Time to query for 200 specific images when the iRODS catalog had 30,000 files – 25 sec

After:

Time to query for specific folders when the iRODS catalog has approx. 3500 Collections – 12 sec

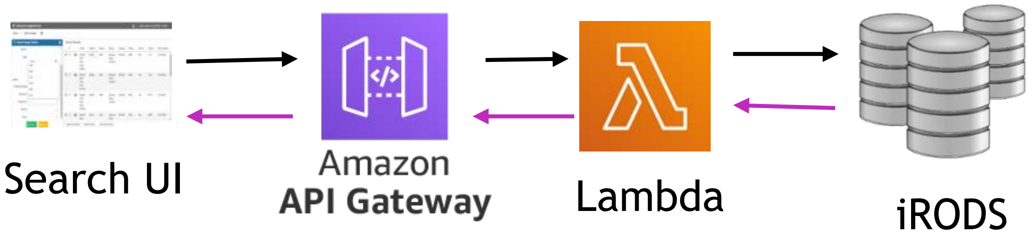
Time to query for the files in the selected Collection with 4200 files – 8 sec

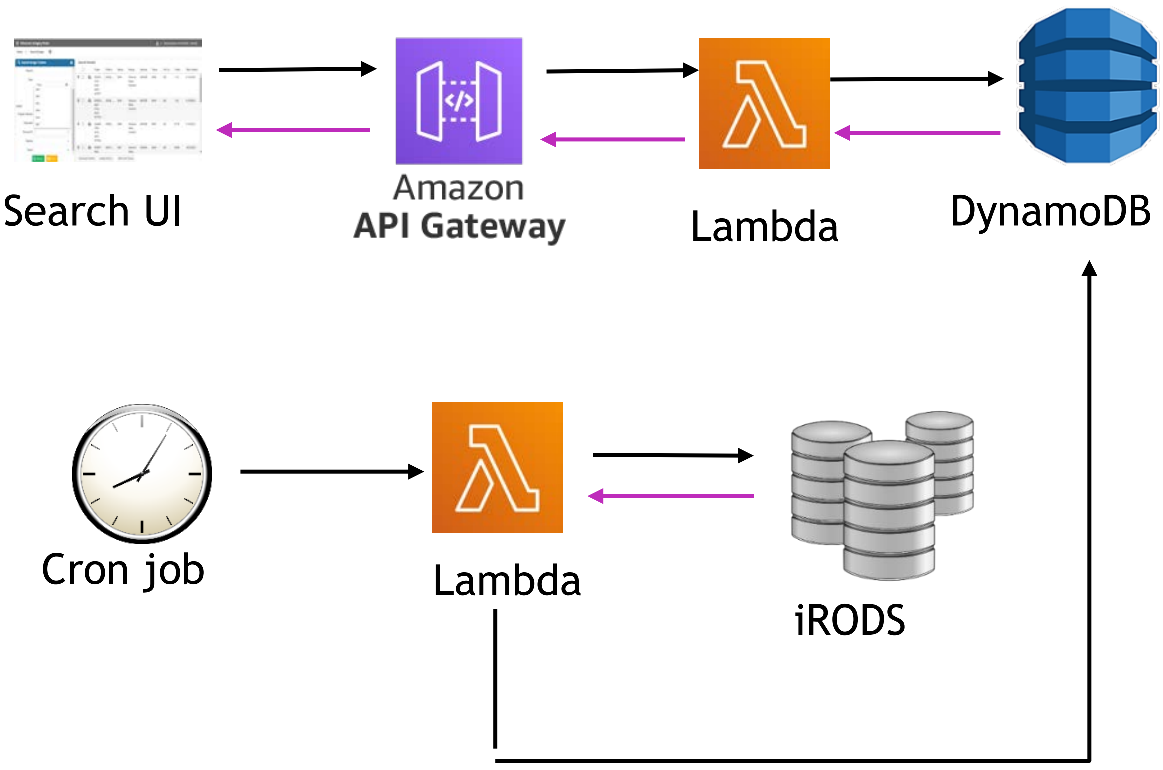
* The time is response time of the Python REST Service that includes creating a data structure after query to IRODS.

Challenges and solutions:

Performance issue to load Search form with unique values

- DynamoDB table has values populated using a CRON job
- Cron job triggers some services to load the data in the table
- Search portal hits a REST services to query the DynamoDB





Performance Metrics *:

Before:

Time to query for unique values of 6 attributes from iRODS – 24 sec

After:

Time to query for unique values of 6 attributes from DynamoDB – 1 sec

* The time is response time of the Python REST Service

Challenges and solutions (continued) :

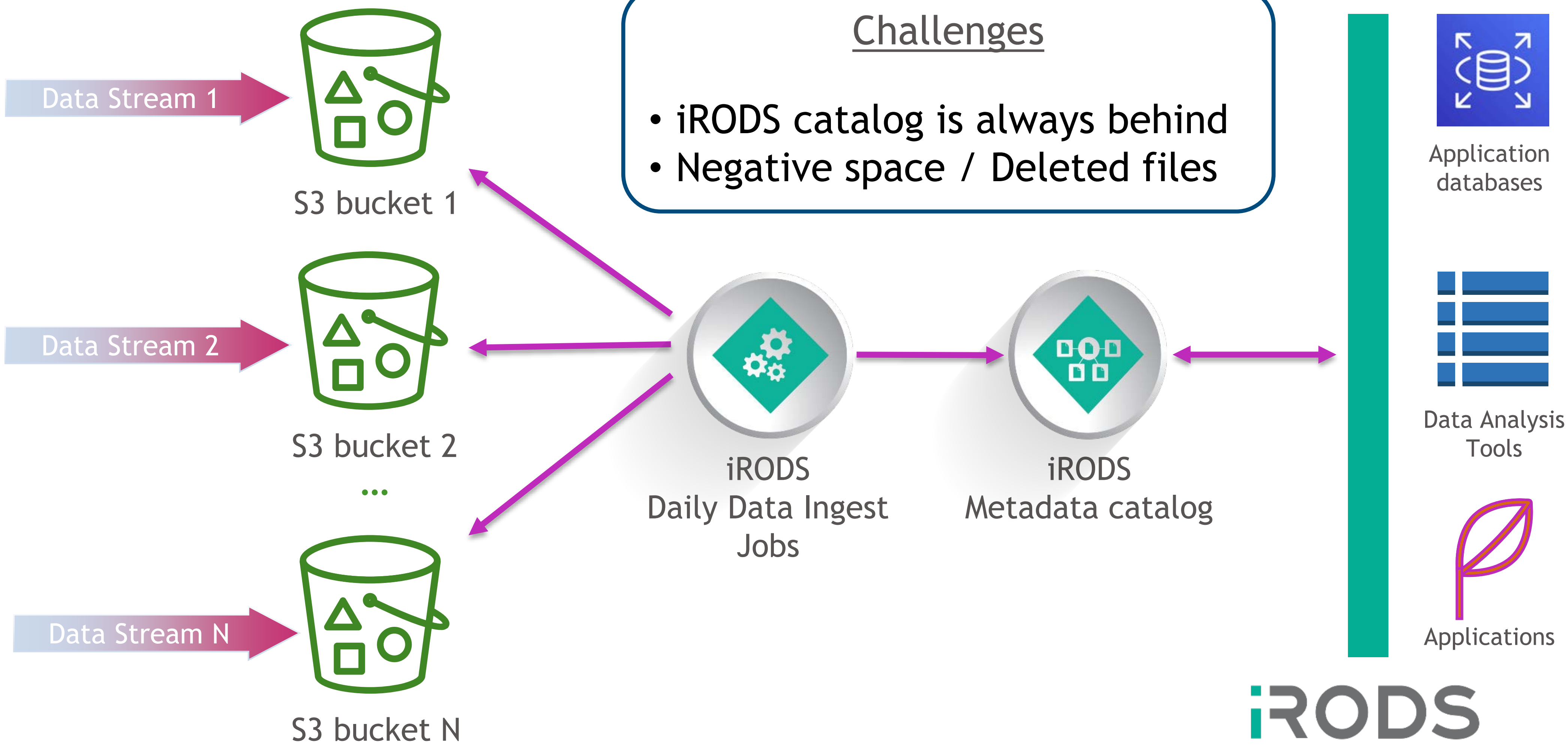
Performance issue to update existing tags for files in iRODS:

- Filtered the list to contain only the attributes that needed to be updated or newly added.
- iRODS client API function msiModAVUMetadata sets one AVU for a data type or collection per transaction.
- Resolved the performance issue using the atomic remove and add functions from the iRODS client API.
- API now has new function msi_atomic_apply_metadata_operations for atomic add or remove of avus.
- Function takes a json object with list of all avus to add or remove.
- Function also takes a data object or collection to set avus on.
- Removed the existing avus and then added the updated values as new avus.
- Performance improved by 3000 times

Performance Metrics *:
Before:
Time to update 25 attributes for 6 files – 50 min
After:
Time to update 5 attributes and add 20 new avus for 6 files – 1.12 sec

* The time includes the other business logic as well.

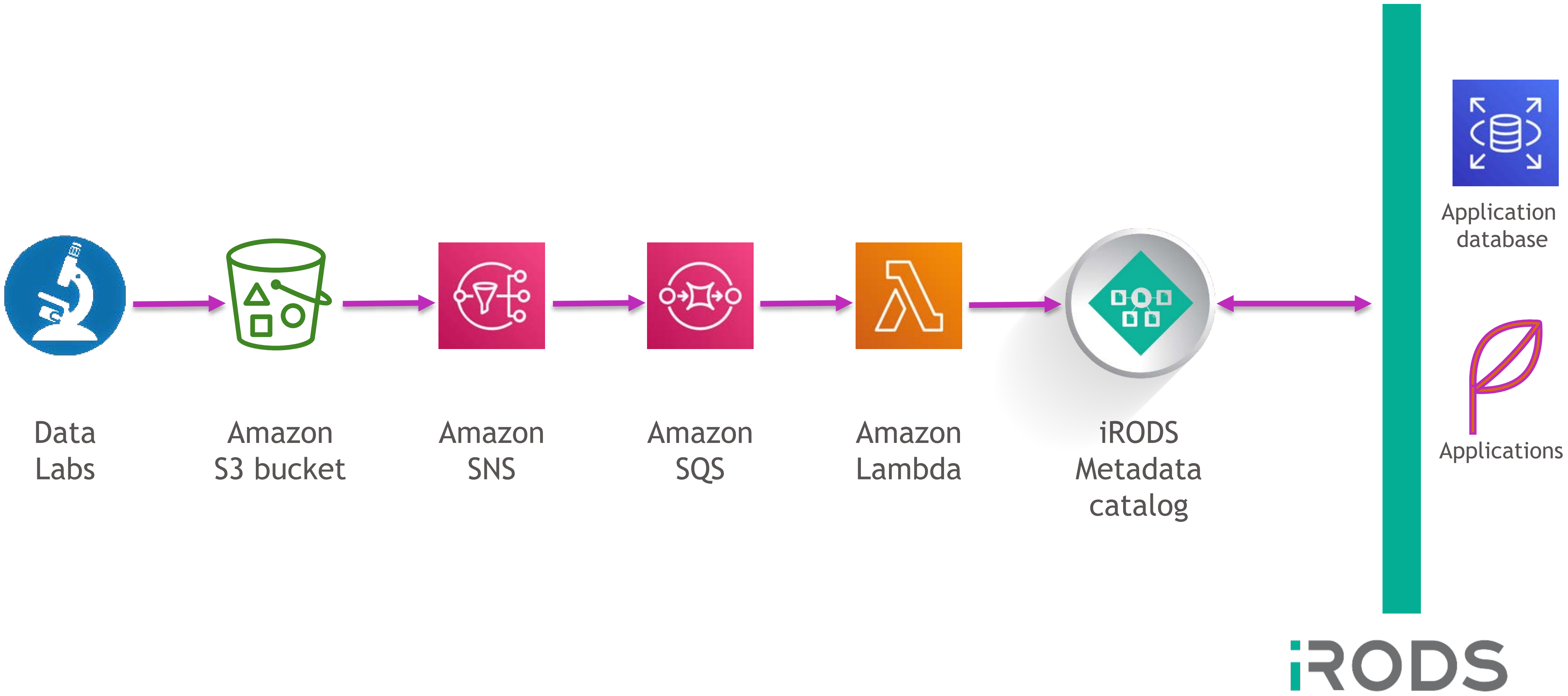
iRODS data ingest - standard approach



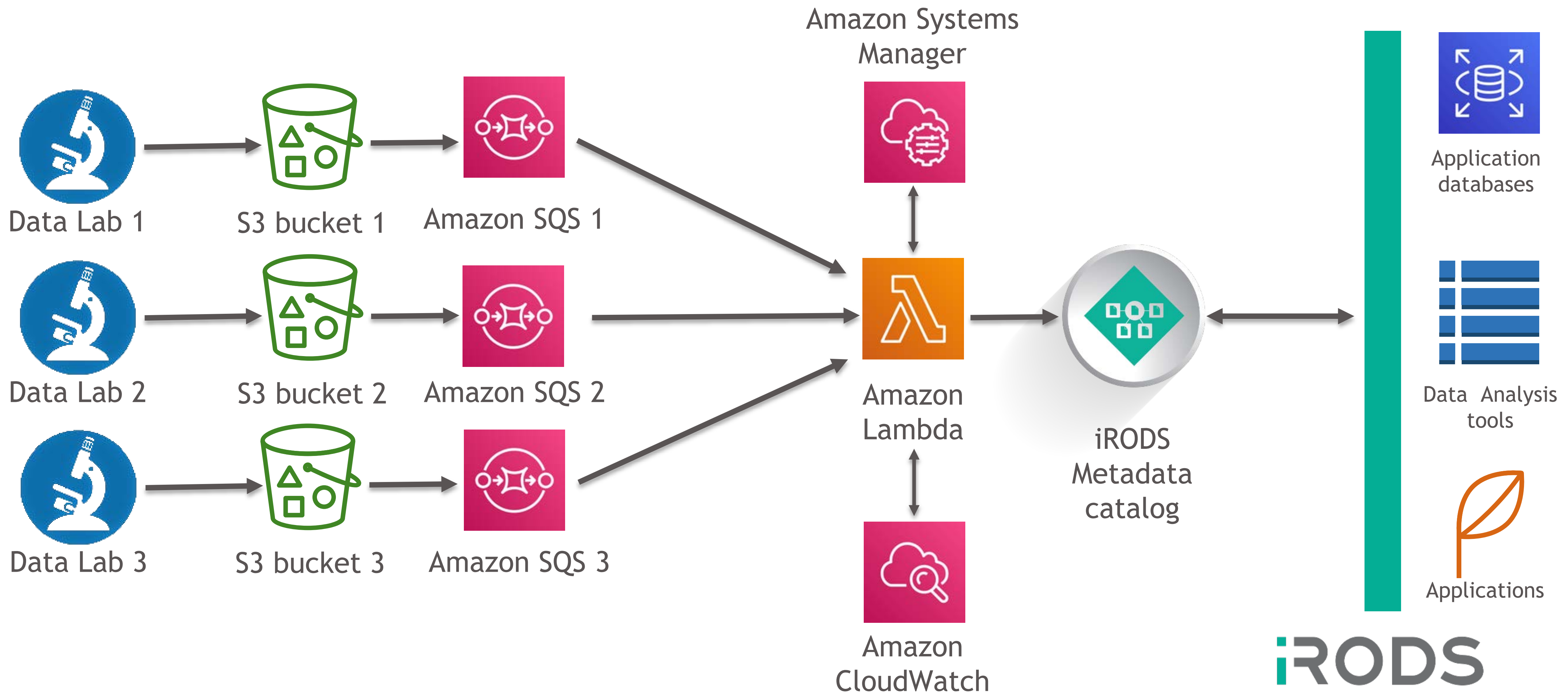
Challenges

- iRODS catalog is always behind
- Negative space / Deleted files

Near real time data ingest - AWS Lambda function



Updating iRODS Catalog with multiple S3 events



iRODS S3 Client AWS Lambda Function

This AWS Lambda function updates an iRODS Catalog with events that occur in multiple S3 buckets. Files created, renamed, or deleted in S3 appear quickly in iRODS.

- Configure trigger on all **ObjectCreated** and **ObjectRemoved** events for a connected S3 bucket.
- Triggers produce messages in AWS SNS/SQS
- Lambda is “listening” to SQS
- The connection information is stored in the Parameter (AWS Systems Manager --> Parameter Store) as a JSON object string.
- SSL Support – also part of **IRODS_ENVIRONMENT_SSM_PARAMETER**
- This Lambda function can be configured to receive events from multiple sources at the same time.
- GitHub repository: https://github.com/irods/irods_client_aws_lambda_s3
- Release 1.2 date: Jul 08, 2020 (Implemented support of multi-part ObjectCreate event)

iRODS S3 Client AWS Lambda Function (continued)

This AWS Lambda function updates an iRODS Catalog with events that occur in multiple S3 buckets. Files created, renamed, or deleted in S3 appear quickly in iRODS.

- iRODS is assumed to have its associated S3 Storage Resource(s) configured with HOST_MODE = cacheless_attached
- If SQS is involved, it is assumed to be configured with batch_size = 1
- Handler: irods_client_aws_lambda_s3.lambda_handler
- Runtime: Python 3.7
- Environment Variables:
  - **IRODS_COLLECTION_PREFIX : /ourZone/home/rods/s3**
  - **IRODS_ENVIRONMENT_SSM_PARAMETER_NAME : irods_default_environment**
  - **IRODS_MULTIBUCKET_SUFFIX : _s3**

Processing Data at Scale and Features

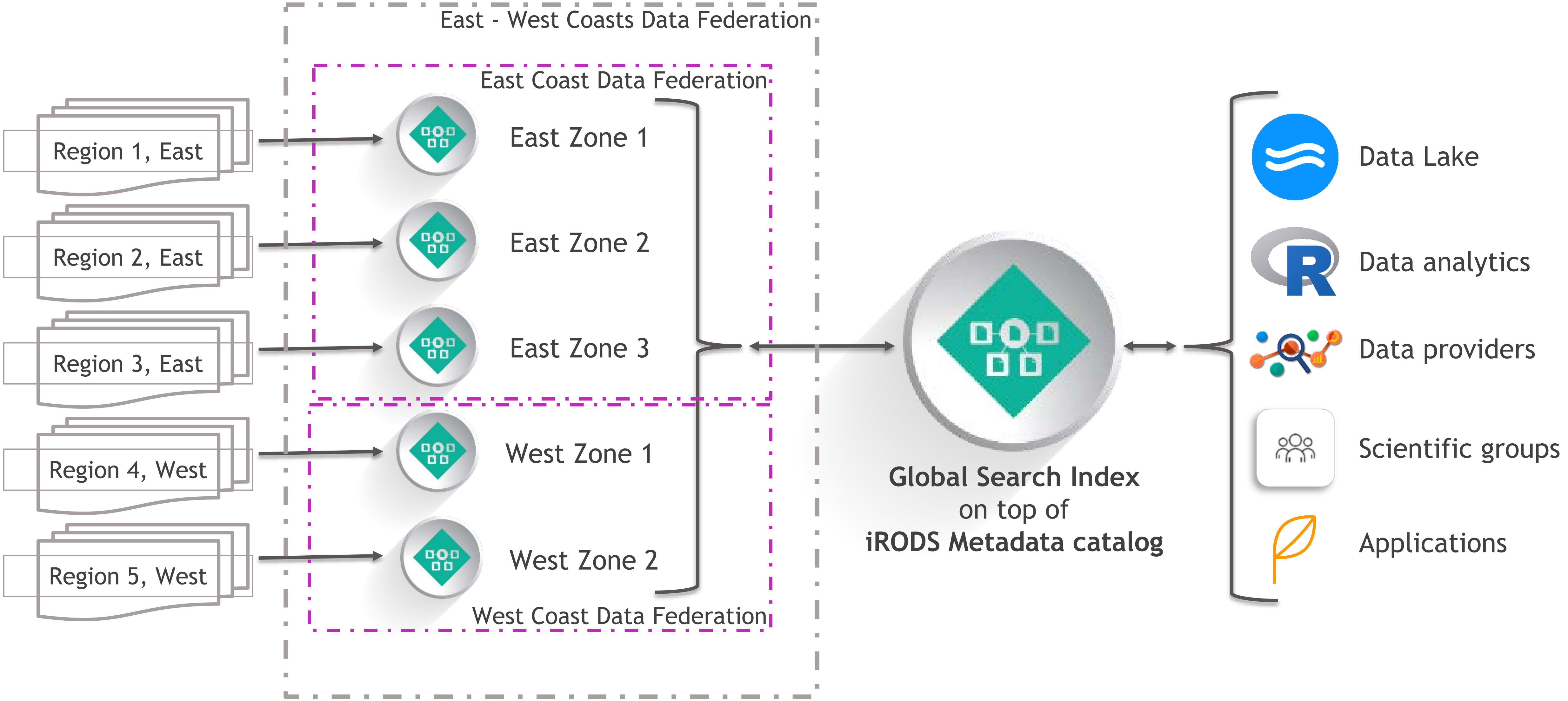
Using iRODS for managing petabytes of data in hundreds of millions of files on distributed storage resources spread across the country.

- Number of S3 buckets: **200+**
- Number of objects in S3: **800+ millions**
- Size of dataset per zone: **~10+ PB**
- Processing rate (regular data ingest): **5 millions objects per hour**

Plugins and features

- S3 / EC2
- NFSRODS
- Metalnx
- AWS RDS PostgreSQL r.12 / Trigram indexes
- AWS Lambda functions over SSL

Towards iRODS Data Farm



Thank you

Radha Konduri | Manager, Data Management & Platforms
Email: radha.konduri@bms.com

Dmitry Khavich | iRODS support, Data Management & Platforms
Email: dmitry.khavich@bms.com

Oleg Moiseyenko | Associate Director, Data Management & Platforms
Email: oleg.moiseyenko@bms.com

Acknowledgements

- iRODS support team
- iRODS / RENCI
- iRODS UGM conference papers
- Open-source community
- DIP team
- IOCT team