New iRODS APIs: Presenting as HTTP and S3

Justin James
Applications Engineer
iRODS Consortium

March 11-13, 2024
CS3 2024
CERN, Geneva, Switzerland

iRODS CONSORTIUM

renci

RESEARCH \ ENGAGEMENT \ INNOVATION

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Our Membership

iRODS

Bibliothèque et Archives nationales Québec

Universiteit Utrecht

What is iRODS

iRODS

Open Source

- C++ client-server architecture
- BSD-3 Licensed, install it today and try before you buy

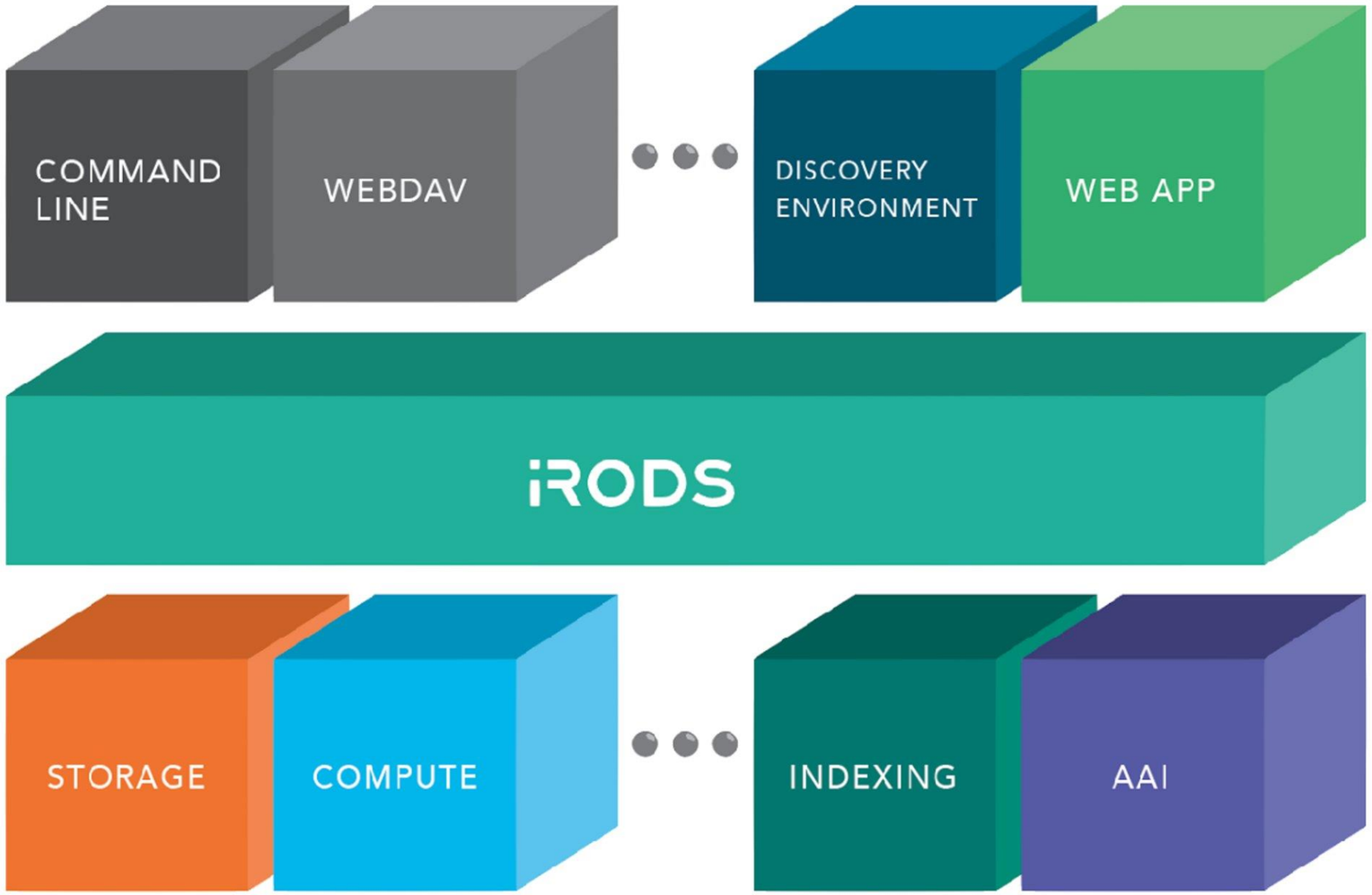
Distributed

- Runs on a laptop, a cluster, on premises or geographically distributed

Data Centric & Metadata Driven

- Insulate both your users and your data from your infrastructure

iRODS as the Integration Layer



iRODS Clients

iRODS provides a layer of abstraction which integrates with your pre-existing infrastructure.

This flexibility allows your infrastructure to continue to change over time.

Existing Infrastructure

Why use iRODS?

People need a solution for:

- Managing large amounts of data across various storage technologies
- Controlling access to data
- Searching their data quickly and efficiently
- Automation

The larger the organization, the more they need software like iRODS.

Protocol Plumbing - Presenting iRODS as other Protocols

Over the last few years, the ecosystem around the iRODS server has continued to expand.

Integration with other types of systems is a valuable way to increase accessibility without teaching existing tools about the iRODS protocol or introducing new tools to users.

With some plumbing, existing tools get the benefit of visibility into an iRODS deployment.

- WebDAV
- FUSE
- HTTP
- NFS
- SFTP
- K8s CSI
- S3

What is the iRODS HTTP API?

A redesign of the iRODS C++ REST API.

Goals of the project ...

- Present a cohesive representation of the iRODS API over the HTTP protocol
  - Simplify development of client-side iRODS applications for new developers
- Maintain performance close to the iCommands
- Remove behavioral differences between different client-side iRODS libraries
  - Will build new language libraries to wrap the HTTP API
  - C, C++, Java, Python, etc.
- Absorbed by the iRODS server if adoption is significant

iRODS HTTP API - Endpoints

Based on concepts and entities defined in iRODS:

/authenticate	/resources
/collections	/rules
/data-objects	/tickets
/info	/users-groups
/query	/zones

Operations are specified via parameters

- Keeps URLs simple (i.e. **no nesting required**)
- Allows new/existing developers to easily find the endpoint of interest

For example

- To modify a user, investigate /users-groups
- To write data to a data object, investigate /data-objects

iRODS HTTP API - Configuration - Top Level

Single file which defines two sections to help administrators understand the options and how they relate to each other.

Modeled after NFSRODS.

```
{
    // Defines HTTP options that affect how the
    // client-facing component of the server behaves.
    "http_server": {
        // ...
    },

    // Defines iRODS connection information.
    "irods_client": {
        // ...
    }
}
```

iRODS HTTP API - Configuration - http_server

```
"http_server": {
    "host": "0.0.0.0",
    "port": 9000,

    "log_level": "info",

    "authentication": {
        "eviction_check_interval_in_seconds": 60,

        "basic": {
            "timeout_in_seconds": 3600
        },

        "openid_connect": { /* ... options ... */ }
    },

    "requests": {
        "threads": 3,
        "max_size_of_request_body_in_bytes": 8388608,
        "timeout_in_seconds": 30
    },

    "background_io": {
        "threads": 6
    }
}
```

iRODS HTTP API - Configuration - irods_client

```
"irods_client": {
    "host": "<string>",
    "port": 1247,
    "zone": "<string>",

    "tls": { /* ... options ... */ },

    "enable_4_2_compatibility": false,

    "proxy_admin_account": {
        "username": "<string>",
        "password": "<string>"
    },

    "connection_pool": {
        "size": 6,
        "refresh_timeout_in_seconds": 600,
        "max_retrievals_before_refresh": 16,
        "refresh_when_resource_changes_detected": true
    },

    "max_number_of_parallel_write_streams": 3,
    "max_number_of_bytes_per_read_operation": 8192,
    "buffer_size_in_bytes_for_write_operations": 8192,
    "max_number_of_rows_per_catalog_query": 15
}
```

iRODS HTTP API - Example - Stat'ing a collection

```
base_url="http://localhost:9000/irods-http-api/0.2.0"
bearer_token=$(curl -sX POST --user 'rods:rods' "$base_url/authenticate")

curl -s -G -H "Authorization: Bearer $bearer_token" \
  "$base_url/collections"                           \
  --data-urlencode 'op=stat'                        \
  --data-urlencode 'lpath=/tempZone/home/rods'      \
  | jq
```


```
{
  "inheritance_enabled": false,
  "irods_response": {
    "status_code": 0
  },
  "modified_at": 1699448576,
  "permissions": [
    {
      "name": "rods",
      "perm": "own",
      "type": "rodsadmin",
      "zone": "tempZone"
    }
  ],
  "registered": true,
  "type": "collection"
}
```

Release v0.2.0

- https://github.com/irods/irods_client_http_api
- January 25, 2024
- Available via Docker Hub

iRODS S3 API - Goals

- Present iRODS as the S3 protocol
  - Multi-user
  - Multi-bucket
- Load Balancer friendly
- Maintainable

iRODS S3 API - History

- iRODS S3 Working Group
  - https://github.com/irods-contrib/irods_working_group_s3
  - Initial email 20210731
  - Suggested four options

iRODS S3 API - History - Options

1. Update and maintain

https://github.com/bioteam/minio-irods-gateway

Go, wrapping iRODS C API. Not going to be maintainable.

2. minio-irods-gateway converts to use

https://github.com/cyverse/go-irodsclient

Pure Go. Limited by lack of multi-user support.

3. Add irods/gateway-irods.go to upstream

https://github.com/minio/minio/tree/master/cmd/gateway

Pure Go, upstream. Limited by above AND MinIO removed support for the gateway.

4. New C++ implementation

https://github.com/irods/irods_client_s3_api

And here we are.

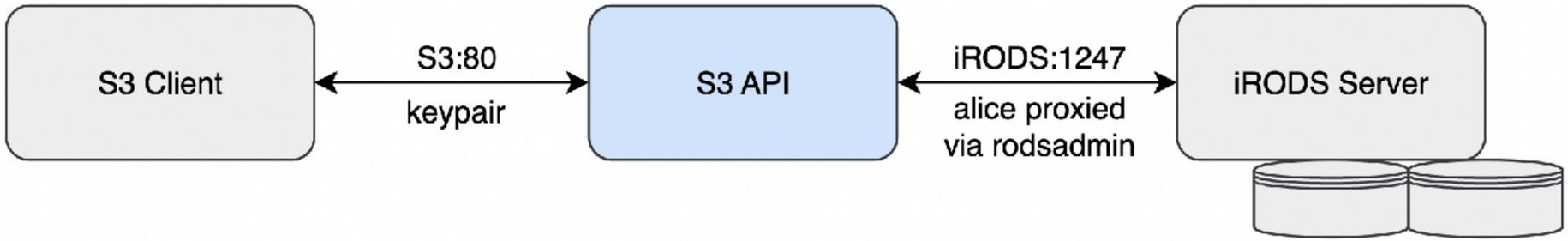
iRODS S3 API - History - Research

- **Option 1** - MinIO with GoRODS (wrapping C)
  - Limited, not maintainable (**Jul 2021**)
- **Option 2** - MinIO with pure go-irodsclient
  - Needed to add (anonymous) ticket functionality
    - Implemented TicketBooth/BoxOffice (**Oct 2021**)
      - But would need admin credentials
      - Might as well just use C++ REST API (**Nov 2021**)
  - Lacks multi-user functionality (**Feb 2022**)
    - Auth code is in MinIO core - gateway code fires 'too late'
- **Option 3** - Get work into upstream MinIO
  - MinIO announced deprecation of gateway (**May 2022**)
    - Too hard / not worth supporting 'legacy' POSIX

- **Option 4** - New C++ Implementation
  - Removes dependency on other codebase(s)
  - 1 collection -> 1 bucket
  - Framework selection (**Aug 2022**)
    - Pistache
    - Oat++
    - Drogon
    - Boost.Beast (**Nov 2022**)
  - Initial endpoints working (**Jan 2023**)
    - User mapping
    - Bucket mapping

- Add S3 protocol support to SFTPGo (**Aug 2022**)
- Searching for existing S3 server in Go (**Sept 2022**)
- Add iRODS backend to Zenko (**Oct 2022**)
- Add JuiceFS frontend to iRODS (**Nov 2022**)
- Add JuiceFS frontend to SFTPGo (**Nov 2022**)
- GarageHQ frontend to iRODS (**Jan 2023**)
- In-memory IBM s3mem-go as inspiration (**Mar 2023**)

iRODS S3 API - Architecture and Status



- Released v0.2.0
- Single binary
- Single configuration file
- Multi-user
- Multi-bucket
- Requires rodsadmin credentials

- Tests passing with:
  - AWS CLI Client
  - Boto3 Python Library
  - MinIO Python Client
  - MinIO CLI Client

iRODS S3 API - Status

- Implemented Endpoints
  - CompleteMultipartUpload
  - CopyObject
  - CreateMultipartUpload
  - DeleteObject
  - DeleteObjects
  - GetBucketLocation
  - GetObject
  - GetObjectLockConfiguration (stub)
  - GetObjectTagging (stub)
  - HeadBucket
  - HeadObject
  - ListBuckets
  - ListObjectsV2
  - PutObject
  - UploadPart

- Investigating
  - ListObjects
  - GetObjectAcl
  - PutObjectAcl
  - PutObjectTagging
  - UploadPartCopy

iRODS S3 API - Configuration

Single file which defines two sections to help administrators understand the options and how they relate to each other.

Modeled after NFSRODS.

```
{
    // Defines S3 options that affect how the
    // client-facing component of the server behaves.
    "s3_server": {
        // ...
    },

    // Defines iRODS connection information.
    "irods_client": {
        // ...
    }
}
```

iRODS S3 API - Configuration - s3_server

```
"s3_server": {
    "host": "0.0.0.0",
    "port": 9000,
    "log_level": "info",
    "plugins": {
        // Each key corresponds to a local shared object file
        "static_bucket_resolver": {
            "name": "static_bucket_resolver",
            "mappings": {
                "<bucket_name>": "/path/to/collection",
                "<another_bucket>": "/path/to/another/collection"
            }
        },
        "static_authentication_resolver": {
            "name": "static_authentication_resolver",
            "users": {
                "<s3_username>": {
                    "username": "<string>",
                    "secret_key": "<string>"
                }
            }
        }
    },
    "region": "us-east-1",
    "authentication": {
        "eviction_check_interval_in_seconds": 60,
        "basic": { "timeout_in_seconds": 3600 }
    },
    "requests": {
        "threads": 3,
        "max_size_of_request_body_in_bytes": 8388608,
        "timeout_in_seconds": 30
    },
    "background_io": { "threads": 6 }
}
```

iRODS S3 API - Configuration - irods_client

```
"irods_client": {
    "host": "<string>",
    "port": 1247,
    "zone": "<string>",

    "tls": { /* ... options ... */ },

    "enable_4_2_compatibility": false,

    "proxy_admin_account": {
        "username": "<string>",
        "password": "<string>"
    },

    "connection_pool": {
        "size": 6,
        "refresh_timeout_in_seconds": 600,
        "max_retrievals_before_refresh": 16,
        "refresh_when_resource_changes_detected": true
    },

    "resource": "<string>",
    "max_number_of_bytes_per_read_operation": 8192,
    "buffer_size_in_bytes_for_write_operations": 8192
}
```

iRODS S3 API - Next Steps

- More Testing
- Additional endpoints
  - Tagging
  - ACLs
- Additional plugins
  - Other bucket mappings
  - Other user mappings

Release v0.2.0

- https://github.com/irods/irods_client_s3_api
- March 6, 2024
- Available via Docker Hub

Questions?

Thank you.