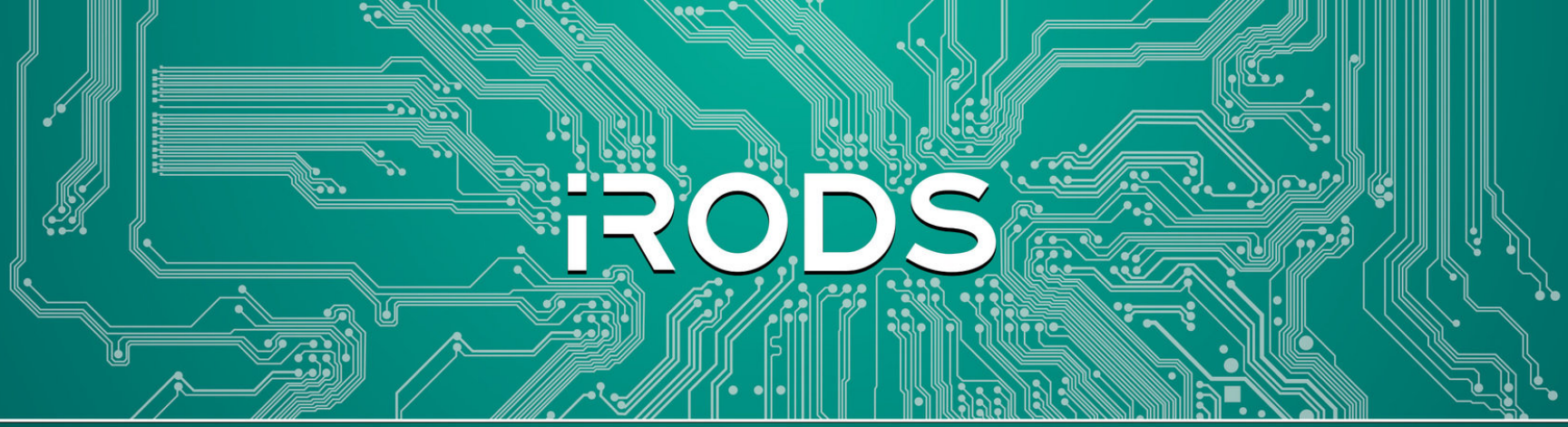


Lustre iRODS Connector

Open Source Data Management for a Parallel File System

Justin James
Applications Engineer, iRODS Consortium

April 26, 2018
Lustre User Group 2018
Argonne National Laboratory

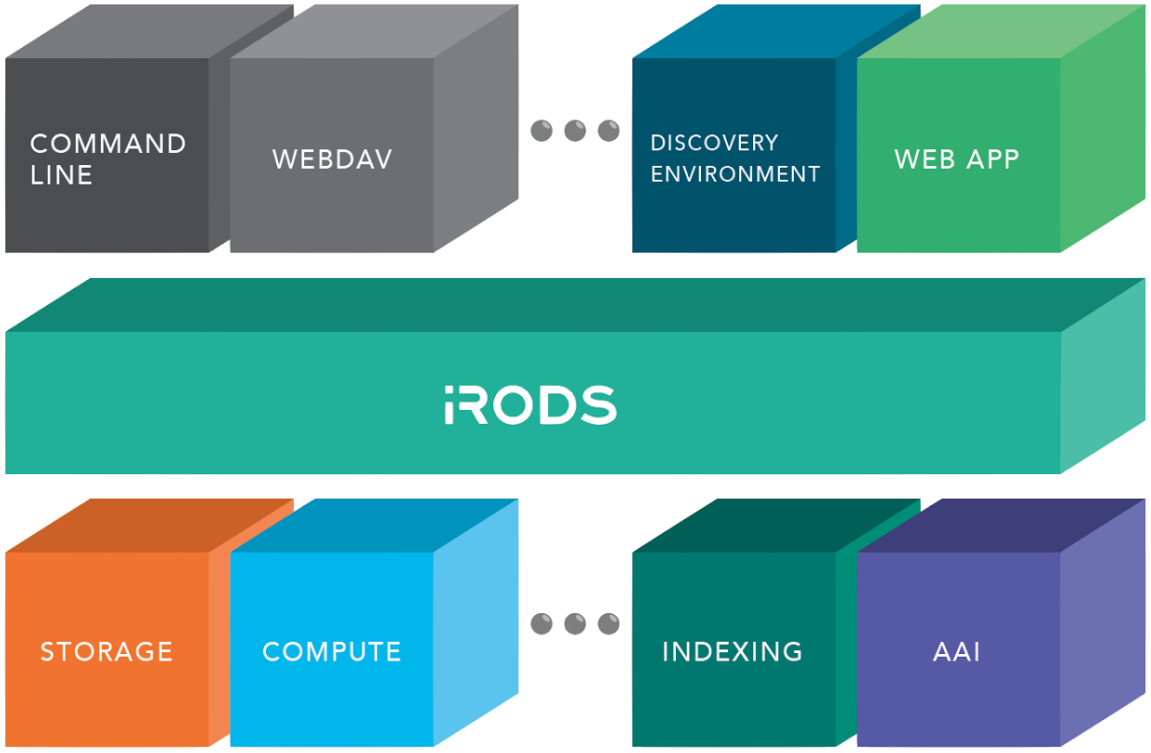
Quick iRODS Overview

iRODS is

- Open source
- Distributed
- Metadata Driven
- Data Centric

A flexible framework for the abstraction of 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

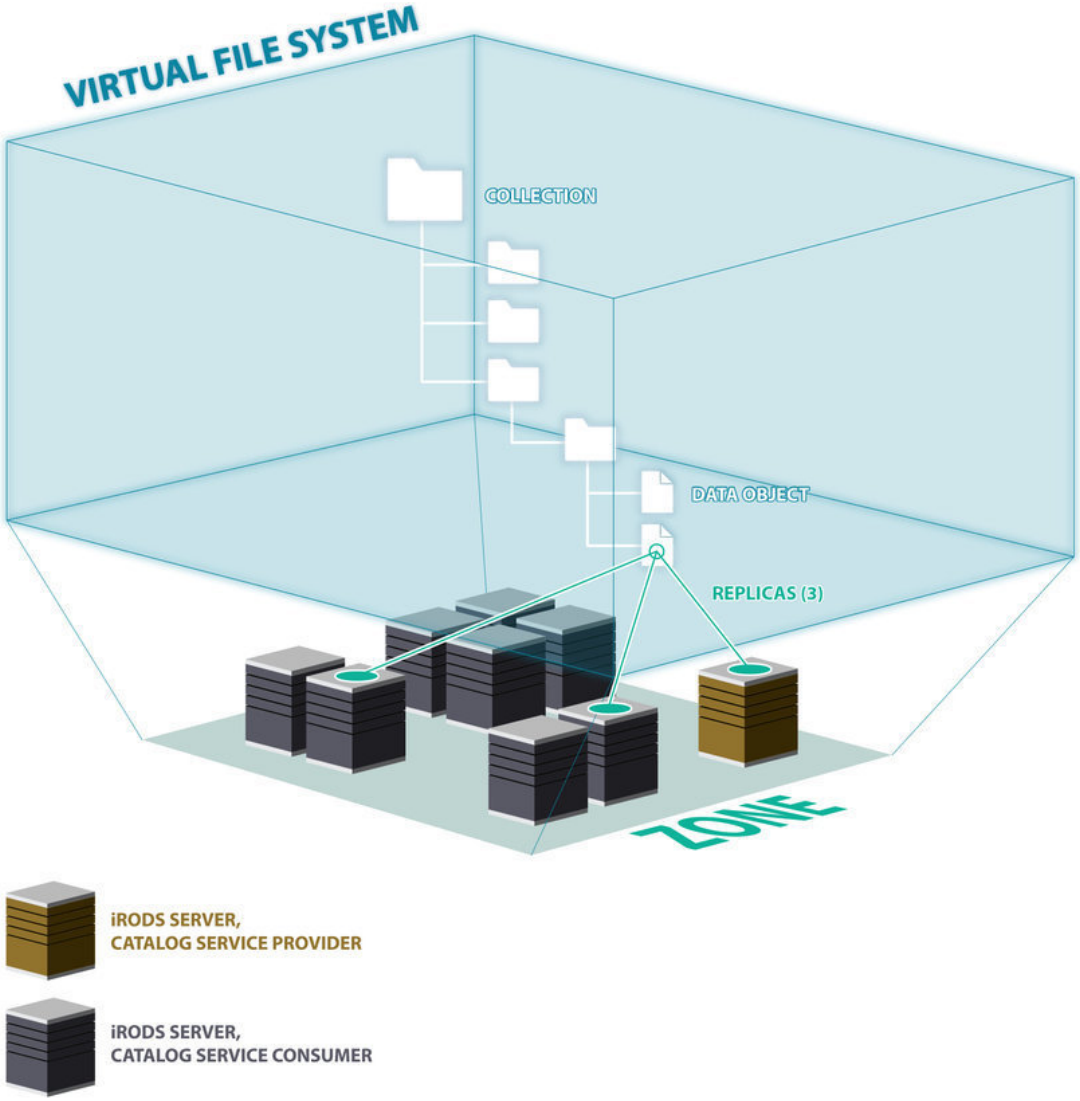
iRODS Overview - Data Virtualization

Combine various distributed storage technologies into a Unified Namespace

- Existing file systems (Lustre, etc.)
- Cloud storage
- On premises object storage
- Archival storage systems

iRODS provides a logical view into the complex physical representation of your data, distributed geographically, and at scale.

iRODS Overview - Data Virtualization



Logical Path

Physical Paths(s)

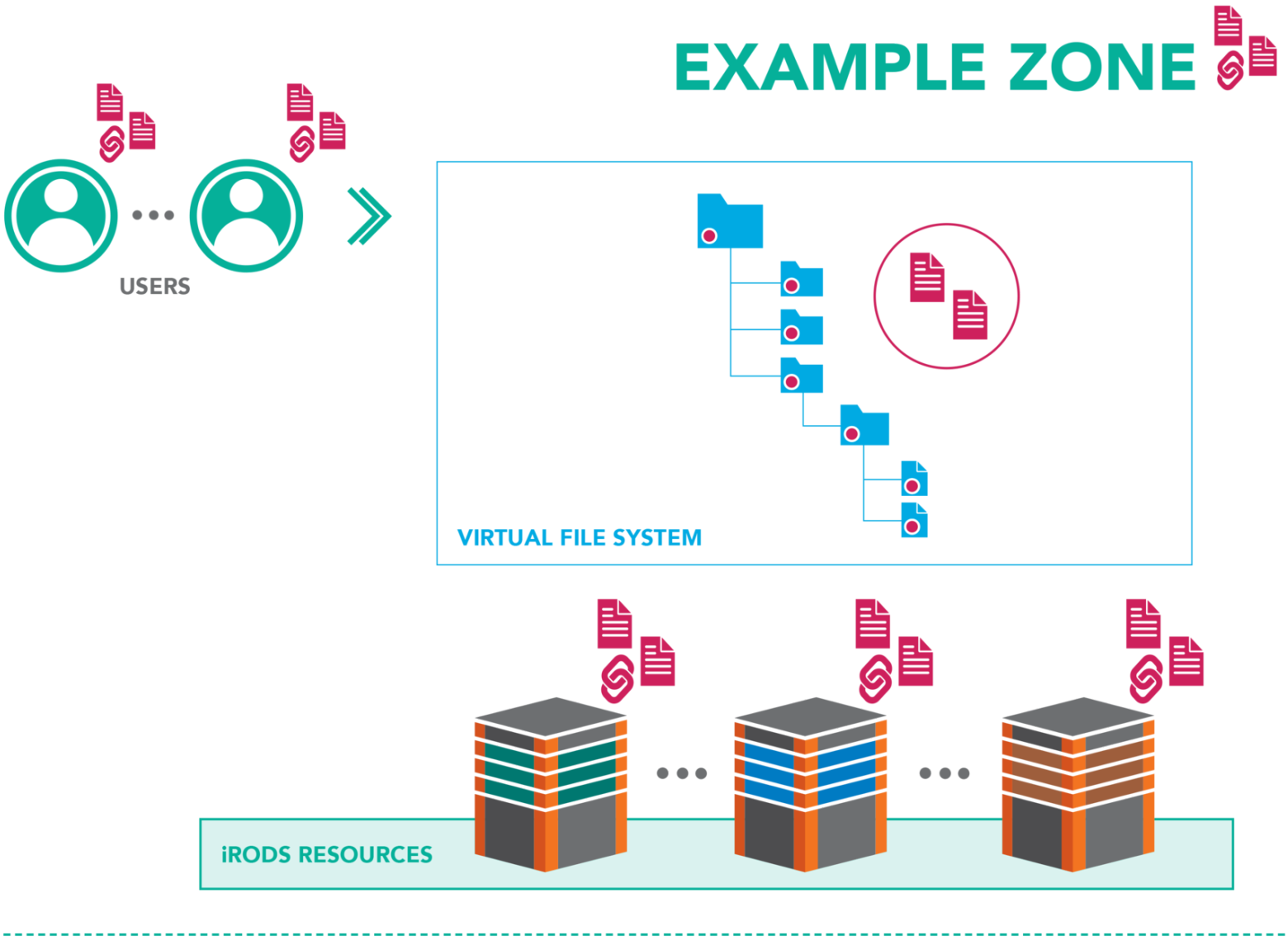
iRODS Overview - Data Discovery

Attach metadata to any first class entity within the iRODS Zone

- Data Objects
- Collections
- Users
- Storage Resources
- The Namespace

iRODS provides automated and user-provided metadata which makes your data and infrastructure more discoverable, operational and valuable.

Metadata Everywhere



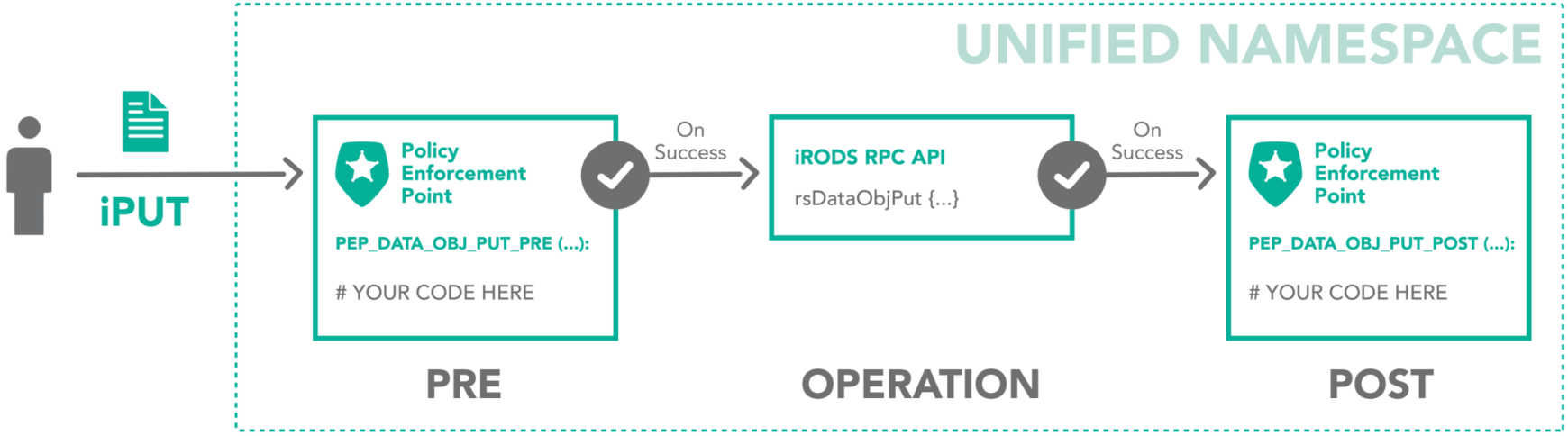
iRODS Overview - Workflow Automation

Integrated scripting language which is triggered by any operation within the framework

- Authentication
- Storage Access
- Database Interaction
- Network Activity
- Extensible RPC API

The iRODS rule engine provides the ability to capture real world policy as computer actionable rules which may allow, deny, or add context to operations within the system.

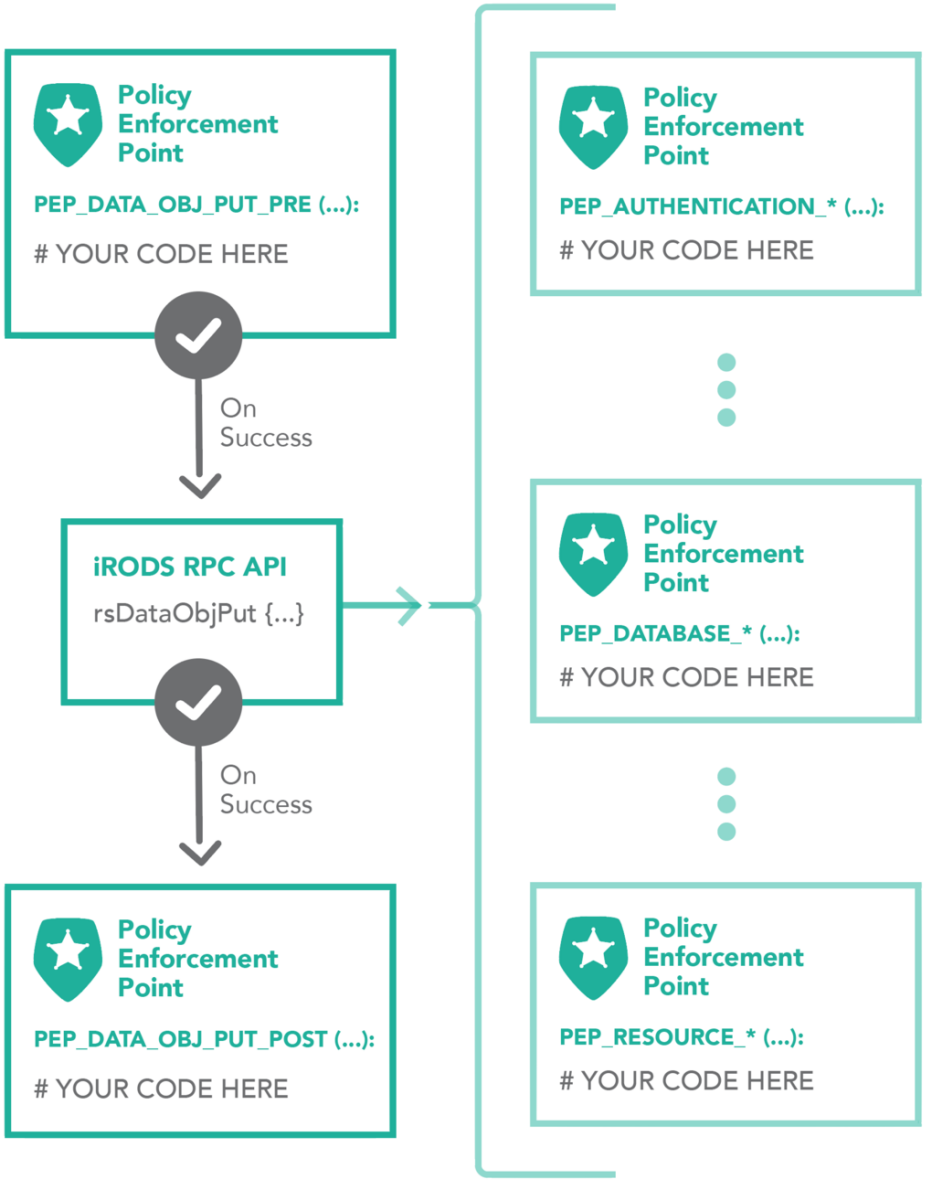
Dynamic Policy Enforcement



The iRODS rule may:

- restrict access
- log for audit and reporting
- provide additional context
- send a notification

Dynamic Policy Enforcement



A single API call expands to many plugin operations all of which may invoke policy enforcement

Plugin Interfaces:

- Authentication
- Database
- Storage
- Network
- Rule Engine
- Microservice
- RPC API

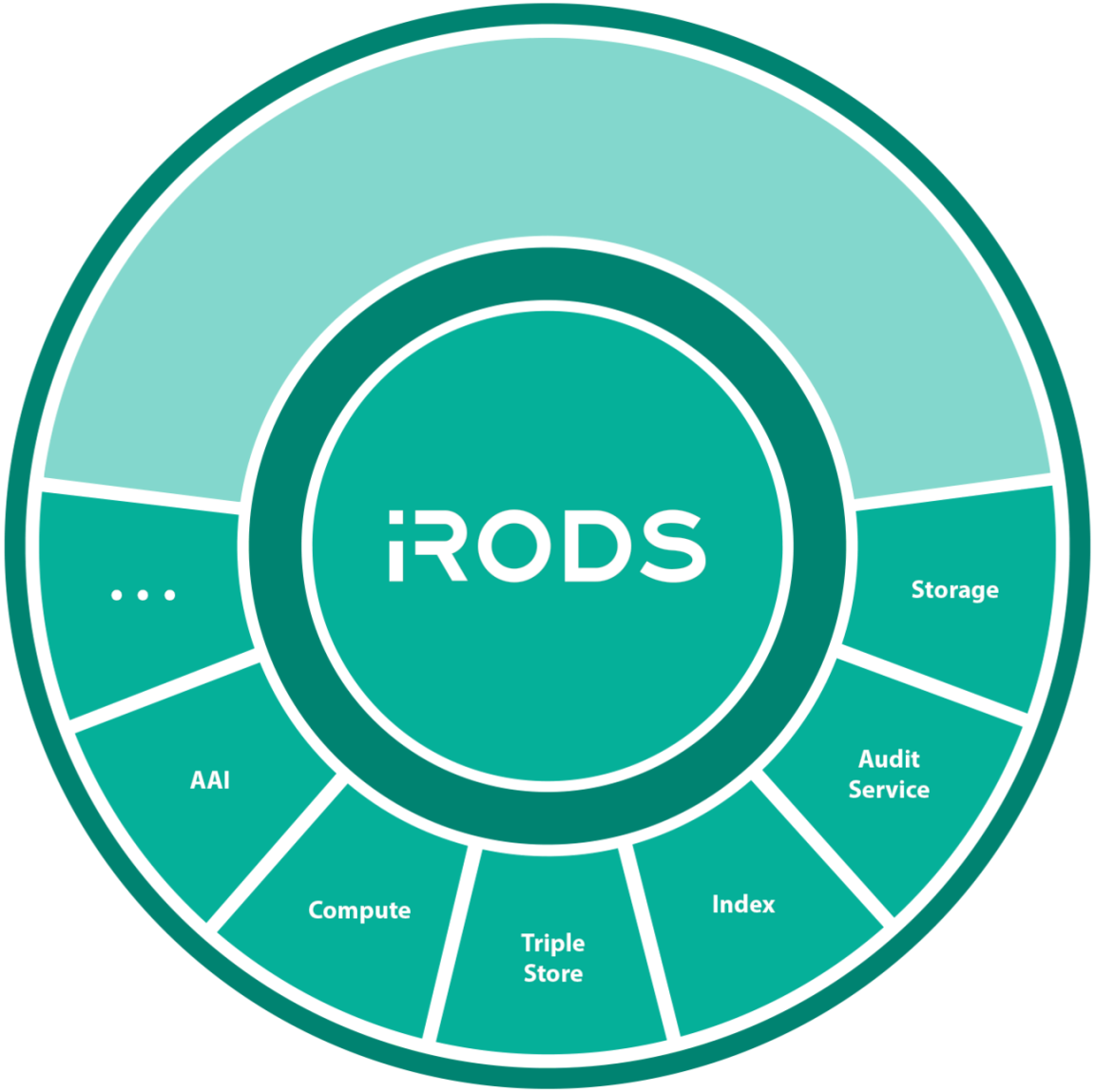
Secure Collaboration

iRODS allows for collaboration across administrative boundaries after deployment

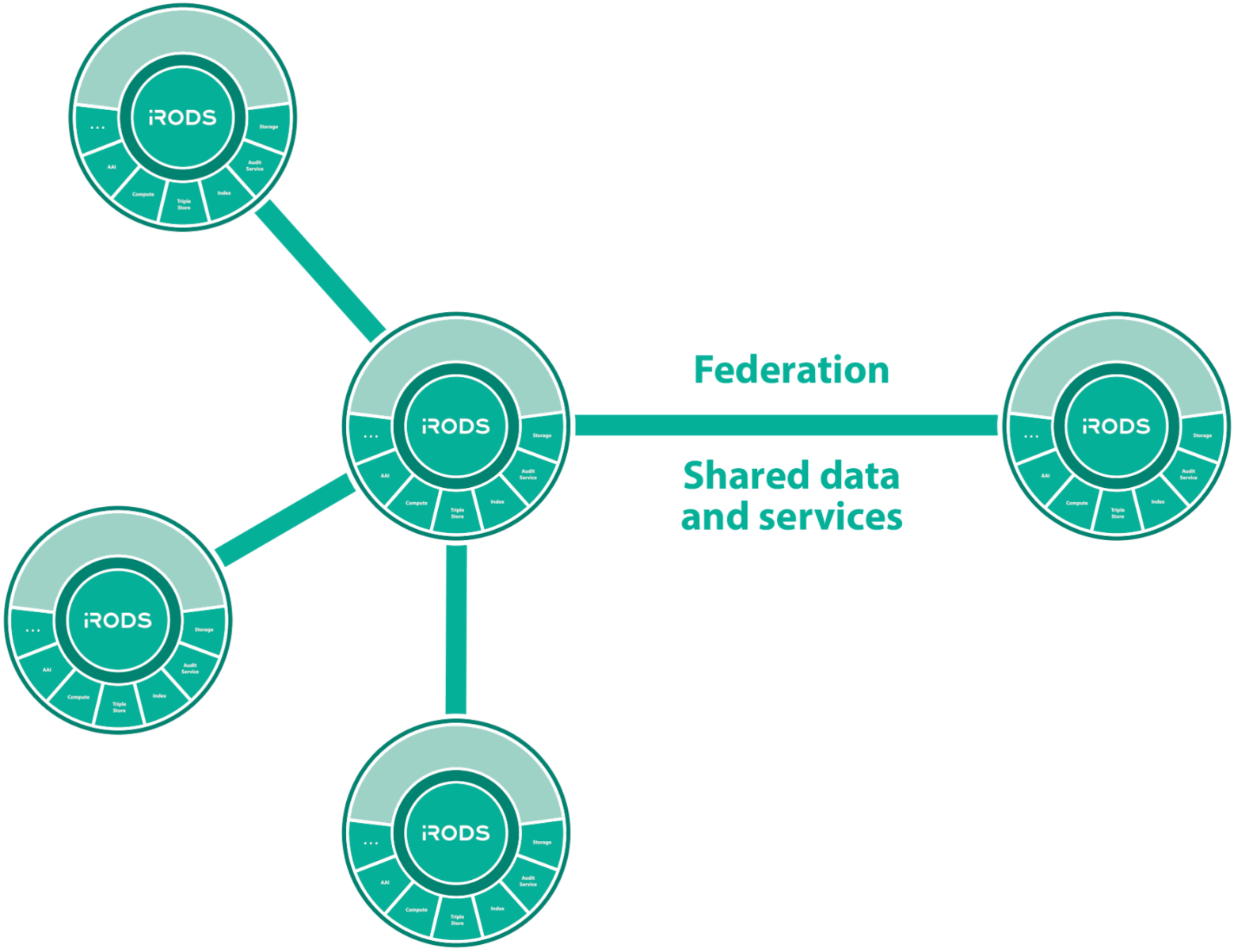
- No need for common infrastructure
- No need for shared funding
- Affords temporary collaborations

iRODS provides the ability to federate namespaces across organizations without pre-coordinated funding or effort.

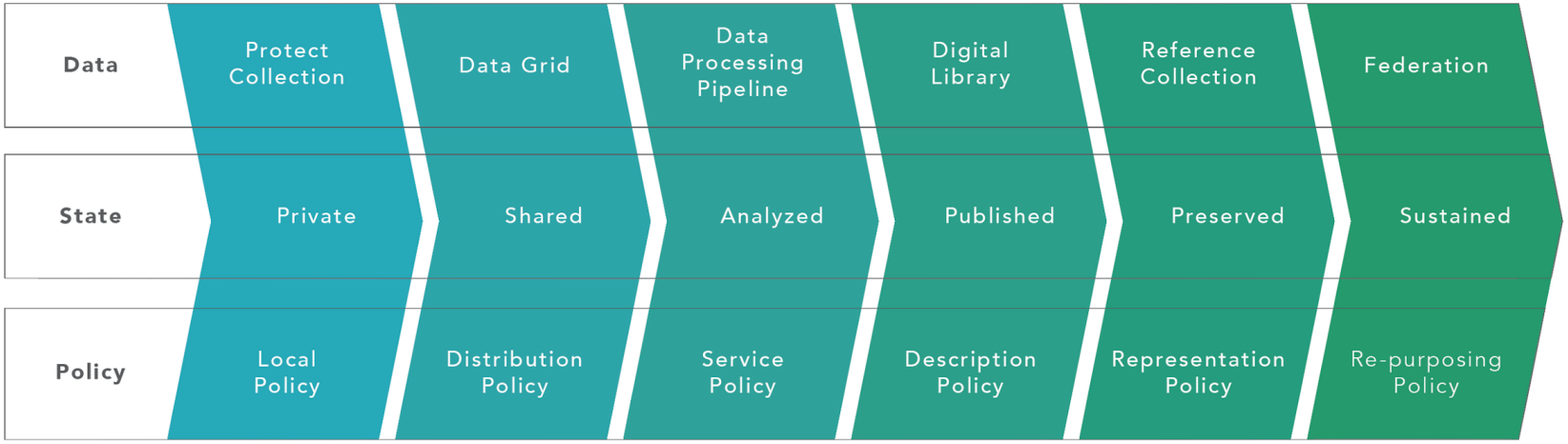
iRODS Service Interface



iRODS Overview - Federation - Shared Data and Services



DATA LIFECYCLE

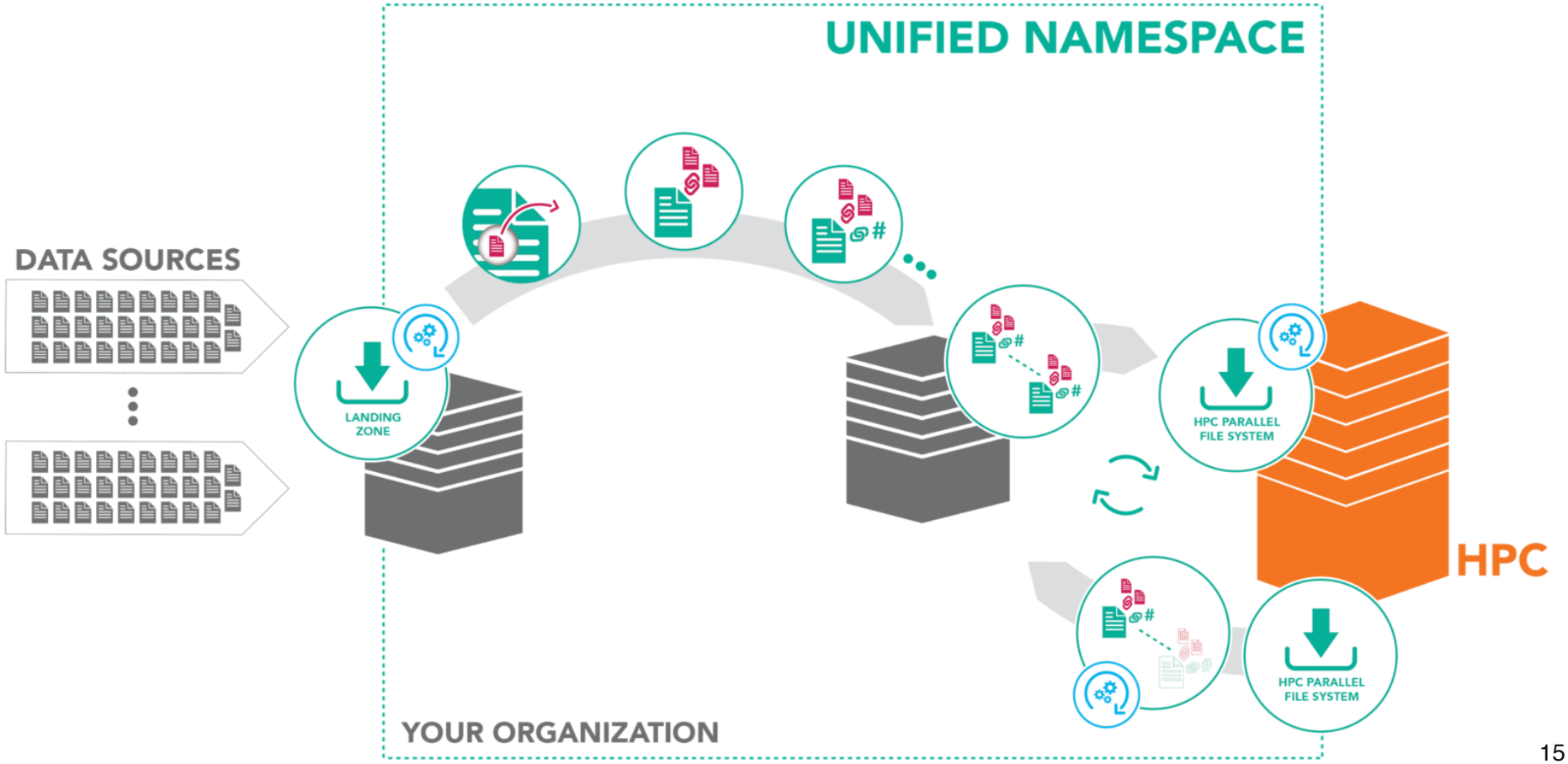


iRODS virtualizes the stages of the data lifecycle through policy evolution

As data matures and reaches a broader community, data management policy must also evolve to meet these additional requirements.

Motivation - Capturing the Products

Focus on the bottom right hand side of the image



iRODS Lustre Framework

iRODS Lustre Framework - Rationale

- Users want iRODS features (user metadata, policies, etc.) while maintaining native Lustre filesystem access for users.

- iRODS already supports POSIX filesystems
  - An iRODS vault can be located within a Lustre mount point

- However, this approach has some drawbacks:
  - All modifications must go through the iRODS API
  - Changes made directly to the filesystem will not be detected by iRODS leading to drift between the filesystem and the iRODS catalog
  - Existing tools must be rewritten to accommodate the iRODS API
  - Does not take advantage of the highly efficient and distributed nature of Lustre

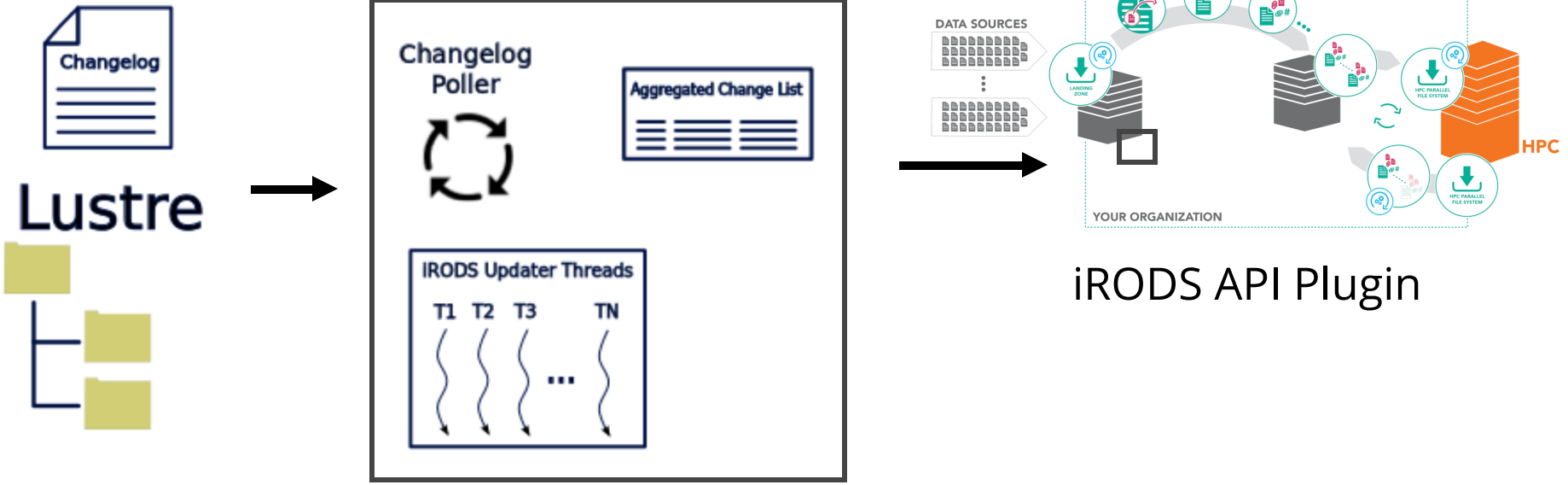
iRODS Lustre Framework - Design

- Add iRODS features on top of the Lustre filesystem
  - Users may continue to directly access filesystem.
  - Use the Lustre changelog to keep iRODS data objects and collections in sync with the Lustre filesystem.
  - All catalog updates are in-place registrations only. No data copies are required.
  - Keep iRODS out of the critical data path. Performance of the Lustre cluster is not impacted.

iRODS Lustre Framework - Efficiency Considerations

- Accumulate and aggregate updates
- Lustre Connector uses multiple threads to send batched updates to iRODS in parallel
- A plugin that resides within iRODS receives and processes the bulk updates
- Support scaling with multiple iRODS endpoints and a distributed database

iRODS Lustre Framework - Overview



iRODS Lustre Framework - iRODS API Plugin

- A dedicated iRODS API plugin within iRODS accepts and processes bulk updates from the iRODS Lustre Connector.

- There are two modes for the iRODS API plugin:

  - Direct - Plugin performs direct database updates. This plugin bypasses the iRODS API architecture. Prioritizes efficiency over policy.

  - Policy - Plugin performs updates via iRODS API. This executes all policy enforcement points (PEPs) within iRODS. Prioritizes policy over speed.

iRODS Lustre Framework - Example

```
[root@zfs1 ~]# mount -t lustre 192.168.56.31@tcp1:/lustre01 /lustre01

[root@zfs1 bld]# echo 'this is a test' > /lustre01/testfile.txt && ils
/tempZone/lustre:
  C- /tempZone/lustre/dir1

[root@zfs1 bld]# ils
/tempZone/lustre:
  testfile.txt
  C- /tempZone/lustre/dir1

[root@zfs1 bld]# iget testfile.txt -
this is a test

[root@zfs1 bld]# imeta add -d testfile.txt description 'just a test file'

[root@zfs1 bld]# imeta ls -d testfile.txt
AVUs defined for dataObj testfile.txt:
attribute: description
value: just a test file
units:
----
attribute: lustre_identifier
value: 0x20000cf21:0x3:0x0
units:

[root@zfs1 bld]# mv testfile.txt dir1

[root@zfs1 bld]# ils dir1
/tempZone/lustre/dir1:
  test4.txt
  testfile.txt
```

iRODS Lustre Framework - Configuration

```
{
        "mdtname": "lustre01-MDT0000",
        "lustre_root_path": "/lustre01",
        "irods_register_path": "/tempZone/lustre",
        "irods_resource_name": "demoResc",
        "irods_api_update_type", "policy",
        "log_level": "LOG_INFO",
        "changelog_poll_interval_seconds": 1,
        "irods_client_connect_failure_retry_seconds": 30,
        "irods_client_broadcast_address": "ipc:///irods_client_broadcast_events",
        "changelog_reader_broadcast_address": "ipc:///changelog_reader_broadcast_events",
        "changelog_reader_push_work_address": "ipc:///changelog_reader_push_work_events",
        "result_accumulator_push_address": "ipc:///result_accumulator_push_address",
        "irods_updater_thread_count": 5,
        "maximum_records_per_update_to_irods": 200,
        "maximum_records_to_receive_from_lustre_changelog": 500,
        "message_receive_timeout_msec": 2000,
        "thread_1_connection_parameters": {
                "irods_host": "localhost",
                "irods_port": "1247"
        },
        "thread_2_connection_parameters": {
                "irods_host": "localhost",
                "irods_port": "1247"
        }
}
```

Notable Configuration Entries

- MDT Name
- Mount Point Available to iRODS
- Registration Point in iRODS
- Multiple iRODS Endpoints - If not defined, use iRODS environment

iRODS Automated Ingest Capability

- Based on Redis for distributed job scheduling

- Designed to solve two particular use cases:
  - Onboarding of existing body of data
    - discovery and registration of files into iRODS catalog
    - on-the-fly metadata extraction
    - does not rely on the Lustre changelog
  - Disaster recovery
    - can be run periodically to sync iRODS catalog with Lustre MDS
    - does not rely on the Lustre changelog

Goal: iRODS - Parallel and Distributed, Front-to-Back

- Parallel Filesystem
- Automated Ingest Capability
- iRODS
  - with multipart, soon
  - with distributed database plugin
- Auditing Framework
- Indexing Framework

iRODS CockroachDB Database Plugin

- CockroachDB is a distributed database which supports an SQL query interface.
- With CockroachDB, iRODS may have multiple catalog (database) providers within an iRODS grid.
- The iRODS Lustre connector allows you to assign different endpoints to different threads.
- When used with CockroachDB, this allows horizontal scaling of processing Lustre updates without encountering a bottleneck at the database.

Copy 1000 files into Lustre

```
# cp -r /1000files /lustre01/
[root@zfs1 lustre01]# ils
/tempZone/lustre:
  C- /tempZone/lustre/1000files
# iquest "select count(DATA_NAME)"
DATA_NAME = 1000
-----------------------------------------------------------
# ils 1000files
/tempZone/lustre/1000files:
  C- /tempZone/lustre/1000files/dir1
  C- /tempZone/lustre/1000files/dir10
  C- /tempZone/lustre/1000files/dir2
  C- /tempZone/lustre/1000files/dir3
  C- /tempZone/lustre/1000files/dir4
  C- /tempZone/lustre/1000files/dir5
  C- /tempZone/lustre/1000files/dir6
  C- /tempZone/lustre/1000files/dir7
  C- /tempZone/lustre/1000files/dir8
  C- /tempZone/lustre/1000files/dir9
# ils 1000files/dir1 | head -10
/tempZone/lustre/1000files/dir1:
  file1
  file10
  file100
  file11
  file12
  file13
  file14
  file15
  file16
```

Enable policy

```
{
        "mdtname": "lustre01-MDT0000",
        "lustre_root_path": "/lustre01",
        "irods_register_path": "/tempZone/lustre",
        "irods_resource_name": "demoResc",
        "irods_api_update_type", "policy",
...
...
```

Enable a rule to extract metadata from image files on registration

```
acPostProcForFilePathReg {

    *err = errormsg(msiExecCmd("read_exif.py", $filePath, "null", "null", "null", *std_out_err), *msg)

    msiGetStdoutInExecCmdOut(*std_out_err,*std_out);
    writeLine("serverLog", "XXXX - STDOUT [*std_out]")

    if(*err != 0) {
        writeLine( "serverLog", "FAILED: [*cmd_opt] [*err] [*msg]" )
        failmsg(*err,*cmd_opt)
    } else {
        msiString2KeyValPair(*std_out, *kvp)
        msiSetKeyValuePairsToObj(*kvp, $objPath, "-d")
    }
}
```

Enable Policy and Use a Rule to Extract Metadata

```
# cp ~/images/stickers.jpg .

# imeta ls -d stickers.jpg | head -20
AVUs defined for dataObj stickers.jpg:
attribute: Image Orientation
value: Horizontal (normal)
units:
----
attribute: EXIF ColorSpace
value: sRGB
units:
----
attribute: lustre_identifier
value: 0x20000cf24:0x14b74:0x0
units:
----
attribute: EXIF FNumber
value: 11/5
units:
----
attribute: GPS GPSDestBearingRef
value: T
units:
```