Integration of iRODS in a distributed storage system through HTTP and Python API

Debaecker Gautier & Pagani Mathia
Integration of iRODS in a Federated IT Service

- IN2P3 Computing center
  - Brief presentation

- Federated IT Service (FITS)
  - Context of the project
  - Classical iRODS commands
  - iRODS at CC-IN2P3

- iRODS access interfaces
  - Python API
  - HTTP API
IN2P3 Computing center
What is CC-IN2P3?

IN2P3:
- one of the 10 CNRS institutes.
- 19 labs dedicated to research in high energy physics, nuclear physics, astroparticles.

CC-IN2P3:
- Computing resources provider for projects supported by IN2P3 (national and international collaborations).
- Resources opened to researchers and engineers working into these collaborations
- 2 computing rooms (2 x 850 m² or 9150 ft²):
  - 2,000 servers.
  - 800 virtual servers.
- 360 Po of storage
Integration of iRODS in a distributed storage system through HTTP and Python API
Context of the project

Introduction to Federated IT Service (FITS) project
Integration of iRODS in a distributed storage system through HTTP and Python API

Pooling computing and storage resources

- Data access must be simple and transparent for the end user
- Choice of iRODS because already well implemented in CC

https://www.fits.cnrs.fr/

"Ce travail a bénéficié d'une aide de l'Etat gérée par l'Agence Nationale de la Recherche au titre du Programme d'Investissements d'Avenir portant la référence ANR-21-ESRE-0009"
CC-IN2P3 iRODS in a few numbers

- 27 zones.

- 110 groups.

- 1236 users:
  - Maximum of 1M connections per day.
  - Maximum of 8M connections per month.

- 450 millions of files.

- 34 PB:
  - Disks: 5 PiB.
  - Tape: 29.2 PiB.
  - Up to +100 TiB daily growing rate.
Needs and Usages

- **Needs**
  - Durable and robust
  - Can be implemented in pipelines/scripts
  - Different way to connect (IAM/Token, User/Password)
  - Accessible to all types of users and platforms (beginners to advanced)
  - Must adapt to users' needs (with installation, web access etc)

- **Usages**
  - “Classic” icommands
  - A python API-based graphical interface
  - A graphical/web interface utilizing HTTP API
“Classic” Icommands

✔ Avantages
✔ Well implemented
✔ Robust
✔ Easily implemented in scripts/pipeline
✔ Official support

⚠️ Disavantages
⚠️ No Identity and Access Manager (IAM) authentication
⚠️ No more official Windows support
⚠️ Can be complicated for beginners
⚠️ No official graphical interface
iRODS access interfaces
Python API-based interfaces
Why ?

- Wrap the main iRODS i-commands
- Facilitate/automate the use of iRODS for new users
- Add the auto-completion for some iRODS i-commands
- Bring iRODS to Windows users

https://github.com/sigau/easy_irods_commands
A Python API-based interface

First thought → wrap inside a python CLI

- Use the icommands
- Keep some information locally (tree structure, metadata use) for autocompletion

✅ Avantages
- Power and speed of icommands
- Easy for beginner that use CLI
- Add simple functionalities (e.g. idush)
- Self installing
- Library build (can be re-uses in other scripts)

⚠️ Disavantages
- Need the icommands (no Windows)
- No graphical interface
- No IAM Identification

Integration of iRODS in a distributed storage system through HTTP and Python API

27/05/2024
A Python API-based graphical interface

Second thought → add a graphical interface

✔ Avantages
  ✔ Very beginner friendly
  ✔ Use of an explorer for data
  ✔ Use the function of the previous script
  ✔ Based on tkinter (no installation need)

⚠ Disavantages
  ⚠ Still need the icommands (no Windows)
  ⚠ Still no IAM Identification
  ⚠ Cannot be implemented in pipeline
A Python API-based graphical interface

Integration of iRODS in a distributed storage system through HTTP and Python API
A Python API-based graphical interface

Third thought → use the Python iRODS API

✓ BeneFITS
✓ No more icommands need
✓ Windows friendly
✓ Mostly as fast as icommands

⚠ Disadvantages
⚠ Still no IAM Identification
iRODS access interfaces

A graphical/web interface using HTTP API
Integration of iRODS in a distributed storage system through HTTP and Python API
User Dashboard:

- Different information
- Access/explore their project collection

Hello IR user

Your info

Projects you belong to

FITS

Your current project: Project 1

Search for Metadata

Back to parent

Size: ... GB

Add metadata

Folders

Files

Integration of iRODS in a distributed storage system through HTTP and Python API
A graphical/web interface using HTTP API

- **User Dashboard:**
  - Get statistics information about their projects

Integration of iRODS in a distributed storage system through HTTP and Python API

27/05/2024
A graphical/web interface using HTTP API

- **Research Infrastructure Admin Dashboard:**
  - Get statistics information about all projects of their team

Integration of iRODS in a distributed storage system through HTTP and Python API
A graphical/web interface using HTTP API

- **Super Admin Dashboard:**
  - Get statistics information about all projects

Integration of iRODS in a distributed storage system through HTTP and Python API
A graphical/web interface using HTTP API

- Authentication through keycloak or Indigo IAM
- Triad portal <-> IAM <-> iRODS
- Integration in a Symfony web portal allows:
  - Easy collection’s tree consultation
  - Get stats at a glance
  - Add/consult metadata
  - Accessible from anywhere + 0 installation
- Problems encountered
  - Problems until 0.3 and “OAuth Protected Resource” (user+password before)
  - What name to mapped with Indigo IAM (personalized with keycloak)
  - Not optimal for sending and downloading mass of data

27/05/2024
Integration of iRODS in a distributed storage system through HTTP and Python API
Federate IT Service project
- iRODS well implemented at CC

Different usage:
- Classical iRODS commands
- Python API:
  - Implementable
  - Graphical interface
  - Windows friendly
- HTTP API:
  - Authentication through IAM
What’s next?

❖ Propose token authentication
  ➢ For icommands
  ➢ For python API

❖ Offer turnkey icommands pipelines for new user

❖ Enriched python script functionalities

❖ Add an adapted way to send data from the web

❖ Add functionalities asked/needed by the user
Thank you for your time!