iRODS

iRODS 4.2: Policy in Your (Favorite) Language

5th National Data Service Consortium Workshop

Chapel Hill, NC - April 5, 2016

Terrell Russell, Ph.D.

@terrellrussell

Senior Data Scientist, iRODS Consortium

Renaissance Computing Institute (RENCI), UNC-Chapel Hill

iRODS

CONSORTIUM

renci

iRODS Consortium

The iRODS Consortium was created to ensure the sustainability of iRODS and to further its adoption and continued evolution. To this end, the Consortium works to standardize the definition, development, and release of iRODS-based data middleware technologies, evangelize iRODS among potential users, promote new advances in iRODS, and expand the adoption of iRODS-based data middleware technologies through the development, release, and support of an open-source, mission-critical, production-level distribution of iRODS.

Current Members:

renci
D·I·C·E
SEAGATE
DDN
NOVARTIS
IBM
Complete GENOMICS
wellcome trust sanger institute
UCL
NASA
EMC²
cleversafe
NICS

Four Major Areas of Deployment

- Health Care & Life Science
- Oil & Gas
- Media & Entertainment
- Archives & Records Management

Open Source Data Management Middleware

- iRODS **enables data discovery** using a metadata catalog that describes every file, every directory, and every storage resource in the data grid.
- iRODS **automates data workflows**, with a rule engine that permits any action to be initiated by any trigger on any server or client in the grid.
- iRODS **enables secure collaboration**, so users only need to log in to their home grid to access data hosted on a remote grid.
- iRODS **implements data virtualization**, allowing access to distributed storage assets under a unified namespace, and freeing organizations from getting locked in to single-vendor storage solutions.

Open Source Data Management Middleware

- iRODS **enables data discovery** using a metadata catalog that describes every file, every directory, and every storage resource in the data grid.
- iRODS **automates data workflows**, with a rule engine that permits any action to be initiated by any trigger on any server or client in the grid.
- iRODS **enables secure collaboration**, so users only need to log in to their home grid to access data hosted on a remote grid.
- iRODS **implements data virtualization**, allowing access to distributed storage assets under a unified namespace, and freeing organizations from getting locked in to single-vendor storage solutions.

Pluggable Rule Engine

- Part of iRODS 4.2, Spring 2016
- Rule Engine Plugins are written in C++
- Allows rules to be written in any language (both interpreted and compiled)
- Multiple rule engines can run concurrently, allowing calls from one language to another

Pluggable Rule Engine

Rule Engine Plugin	LOC (w/ comments)
iRODS Rule Language	253
Javascript	244
Python	252
Auditing (C++)	157
Default Policy (C++)	492

Defined operations:

- start
- stop
- rule_exists
- exec_rule
- exec_rule_text
- exec_rule_expression

Policy Enforcement Points (PEPs)

Every operation in the entire system is made available as a policy hook.

Some examples include:

audit_pep_api_data_obj_put_post
audit_pep_api_data_obj_put_pre
audit_pep_api_mod_avu_metadata_post
audit_pep_api_mod_avu_metadata_pre
audit_pep_api_reg_replica_post
audit_pep_api_reg_replica_pre
audit_pep_api_ssl_end_post
audit_pep_api_ssl_end_pre
audit_pep_auth_agent_auth_response_post
audit_pep_auth_agent_auth_response_pre
audit_pep_auth_agent_start_post
audit_pep_auth_agent_start_pre
audit_pep_database_check_auth_post
audit_pep_database_check_auth_pre
audit_pep_database_gen_query_post
audit_pep_database_gen_query_pre
audit_pep_database_mod_data_obj_meta_post
audit_pep_database_mod_data_obj_meta_pre
audit_pep_database_reg_data_obj_post
audit_pep_database_reg_data_obj_pre
audit_pep_database_set_quota_post
audit_pep_database_set_quota_pre

audit_pep_exec_microservice_post
audit_pep_exec_microservice_pre
audit_pep_exec_rule_post
audit_pep_exec_rule_pre
audit_pep_network_agent_start_post
audit_pep_network_agent_start_pre
audit_pep_network_client_stop_post
audit_pep_network_client_stop_pre
audit_pep_network_read_header_post
audit_pep_network_read_header_pre
audit_pep_resource_modified_post
audit_pep_resource_modified_pre
audit_pep_resource_open_post
audit_pep_resource_open_pre
audit_pep_resource_rebalance_post
audit_pep_resource_rebalance_pre
audit_pep_resource_resolve_hierarchy_post
audit_pep_resource_resolve_hierarchy_pre
audit_pep_resource_stat_post
audit_pep_resource_stat_pre
audit_pep_resource_write_post
audit_pep_resource_write_pre

Three Rule Bases

iRODS Rule Language

```
# existing iRODS Rule Language - custom.re

irodsFunc(*foo) {
    writeLine("serverLog", "custom.re - BEGIN - irodsFunc(foo): [*foo]");
    pyFunc("called from custom.re");
    writeLine("serverLog", "custom.re - END   - irodsFunc(foo)");
}

getSessionVar(*name, *output) {
    *output = eval("str($"+++*name++")");
}
```

Javascript

```
/* Javascript - core.js */

function jsFunc(foo, callback) {
    callback.writeLine("serverLog", "JAVASCRIPT - BEGIN - jsFunc(foo, callback)");
    callback.writeLine("serverLog", "  - with parameter foo[" + foo + "]");
    try {
        callback.doesnotexist("nope");
    } catch(e) {
        throw e + " -- ERROR HANDLING FTW!";
    }
    callback.writeLine("serverLog", "JAVASCRIPT - END   - jsFunc(foo, callback)");
}
```

Three Rule Bases

Python

```
# Python - core.py

import datetime

def pyFunc(rule_args, callback):
    callback.writeLine('serverLog', 'PYTHON - BEGIN - pyFunc(rule_args, callback)')
    for arg in (rule_args):
        callback.writeLine('serverLog', 'PYTHON -- arg=[' + arg + ']')
    callback.writeLine('serverLog', 'PYTHON - END   - pyFunc(rule_args, callback)')

########################################
# DEMO - Parameters and Error Handling #
########################################
def acPostProcForPut(rule_args, callback):
    callback.writeLine('serverLog', 'PYTHON - BEGIN - acPostProcForPut()')
    callback.irodsFunc("called from python, apples")
    callback.jsFunc("called from python, bananas")
    session_vars = ['userNameClient', 'dataSize',]
    for s in session_vars:
        v = callback.getSessionVar(s, 'dummy')[1]
        callback.writeLine('serverLog', s + ' :: ' + v)
    callback.writeLine('serverLog', 'PYTHON - END   - acPostProcForPut()')
```

DEMO - Parameters and Error Handling

iput a file into iRODS

```
$ iput puppies.jpg
```

rodsLog:

```
Apr  4 13:02:01 pid:26540 NOTICE: Agent process 30993 started for puser=rods and cuser=rods fro
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = PYTHON - BEGIN - acPostProcForPut()
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = custom.re - BEGIN - irodsFunc(foo): [ca
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = PYTHON - BEGIN - pyFunc(rule_args, call
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = PYTHON -- arg=[called from custom.re]
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = PYTHON - END   - pyFunc(rule_args, call
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = custom.re - END   - irodsFunc(foo)
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = JAVASCRIPT - BEGIN - jsFunc(foo, callba
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString =   - with parameter foo[called from pyth
Apr  4 13:02:01 pid:30993 ERROR: [-]    iRODS/server/re/src/rules.cpp:674:int actionTableLookUp
        [-]     iRODS/server/re/src/irods_ms_plugin.cpp:110:irods::error irods::load_microservi
                [-]     iRODS/lib/core/include/irods_load_plugin.hpp:145:irods::error irods::lo
Apr  4 13:02:01 pid:30993 ERROR: -1102000 -- ERROR HANDLING FTW!
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = userNameClient :: rods
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = dataSize :: 95891
Apr  4 13:02:01 pid:30993 NOTICE: writeLine: inString = PYTHON - END   - acPostProcForPut()
Apr  4 13:02:01 pid:30993 NOTICE: readAndProcClientMsg: received disconnect msg from client
Apr  4 13:02:01 pid:30993 NOTICE: Agent exiting with status = 0
Apr  4 13:02:01 pid:26540 NOTICE: Agent process 30993 exited with status 0
```

Discussion

- Powerful abstraction
- Invites new developers
- Provides migration path
- Requires new documentation
- Requires broader security model
- Requires careful consideration

Questions?

Hao Xu, Jason Coposky, Ben Keller, Terrell Russell (2015).
Pluggable Rule Engine Architecture.
iRODS User Group Meeting 2015 Proceedings, pp. 29-34
http://irods.org/wp-content/uploads/2015/09/UMG2015_P.pdf

irods.org
github.com/irods
@irods

Terrell Russell
@terrellrussell