iRODS – Advanced user training

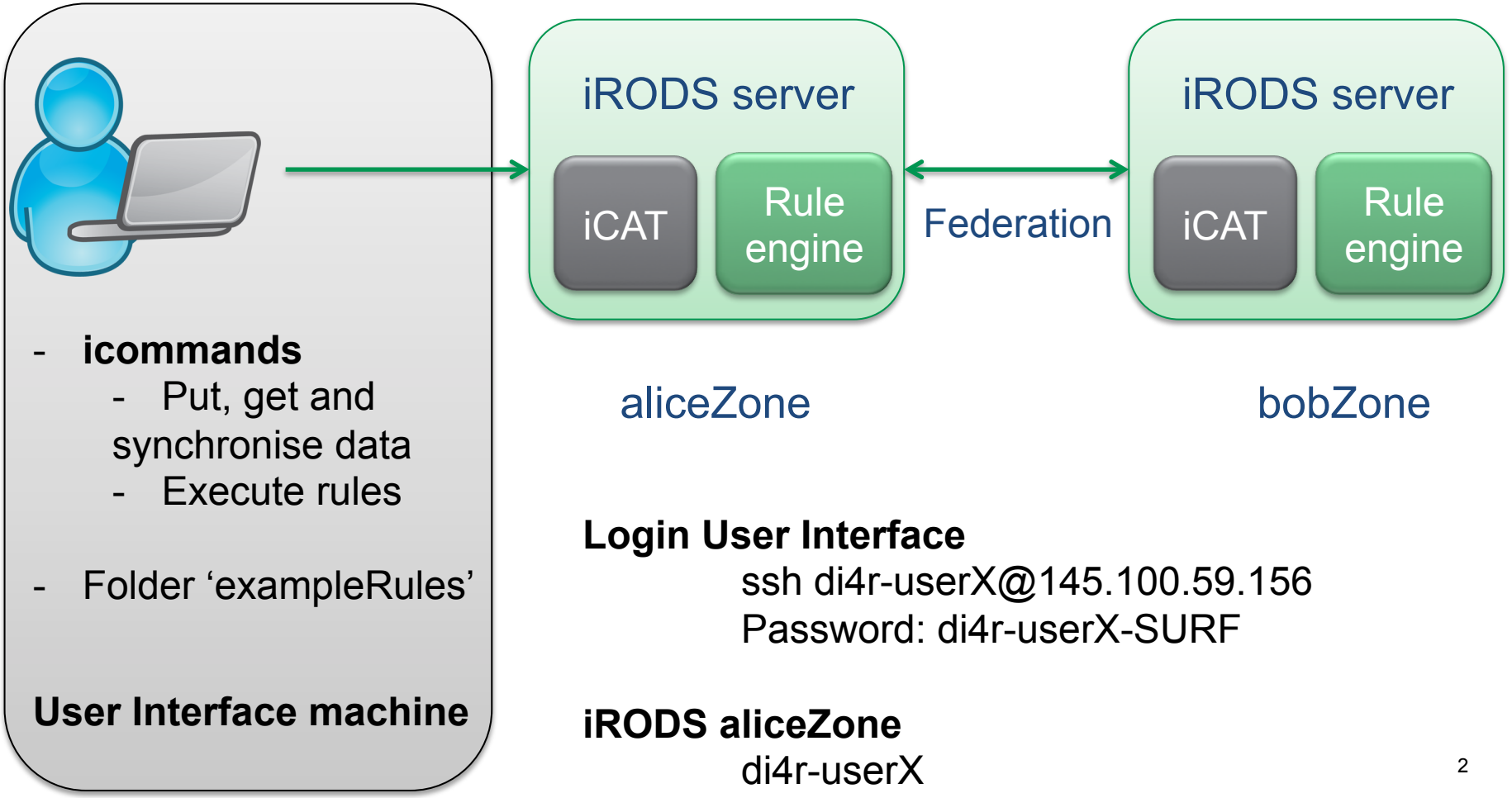
FEDERATIONS AND RULES – S4R WORKSHOP

Christine Staiger

EUDAT

SURF SARA

Training Setup



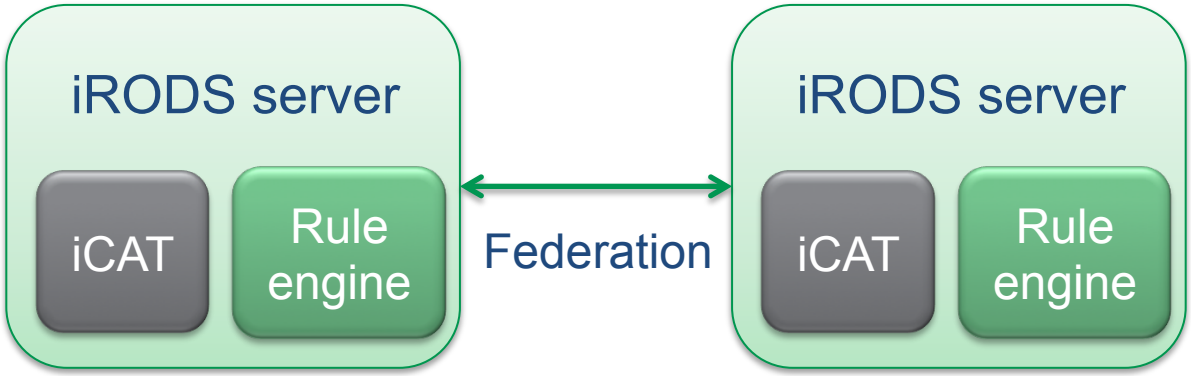
Login User Interface

ssh di4r-userX@145.100.59.156
Password: di4r-userX-SURF

iRODS aliceZone

di4r-userX
Password: di4r-userX-SURF

iRODS Federations



- Two independent iRODS zones, own rule engine and different rulebases
- Federation on system level
- iRODS admins give access to certain users

User

- Authentication with home iRODS zone
- If acknowledges user: Access to federated zone
  /otherIRODSzone/home/user#homeIRODSzone

Rules and micro services

iRODS micro services

- Define actions on data, resources and users → atomic
- C++ functions, calling external libraries
- Used and combined to workflows and policies in iRODS rules

- Predefined microservices
  http://docs.irods.org/4.1.10/doxygen
- Example: msiCollRsync → synchronises two iRODS collections from different zones

- Own micro services:
  - Written in C++
  - Need to be installed on the iRODS server → sysadmin & iRODS admin rights
  - Example: Automatic metadata extraction from HDF5 files

iRODS rules

- iRODS rule engine → built-in interpreter for own language
- Automatise data management tasks
- Standard set of pre-implemented rules constitutes default data policies

- Trigger execution of rules by
  - irule → User
  - Delayed or scheduled execution → User & iRODS admin
  - Actions and policy enforcement points extending and overlaying the default rule base → sysadmin

```
HelloWorld{
        writeLine("stdout", "Hello *name!");
}
INPUT *name="World"
OUTPUT ruleExecOut, *name
```

iRODS standard data policies

- Actions are triggered by client interaction

  - Triggered by client interaction e.g. create new data (iput)
  - acPostProcForPut - Rule for post processing the put operation.

```
acPostProcForPut {msiSysChksumDataObj;
        msiSysReplDataObj("demoResc","all"); }
```

- Policy enforcement points (PEPs) are triggered by any interaction

```
pep_api_data_obj_put_post(
    *COMM, *DATAOBJINP, *BUFFER, *PORTAL_OPR_OUT)
    { acPostProcForPut; }
```

Extending the standard core.re

- Predefined core.re and also pretty empty in standard setup
- Place your (carefully tested) rules directly into core.re
  → bad idea
- Write an own policy.re and configure server
  "re_rulebase_set":[{"filename":”policy”}, {"filename":"core”}]

  → policy.re and core.re build the rule set for this iRODS instance
  → Order matters

Rules: Order matters

- No namespaces!
- First rule that matches (name and variables) will be executed
- Actions and PEPs follow the syntax of rules

Workflow for developing policies/rules

- Write a local rule as iRODS user → irule <file>
  → Debugging

- Put rule on top of all rules in the configured rule set
  → Does it still work?
  → Which rules does it inhibit from being executed

- Bit by bit find the right spot of the rule in the rule base

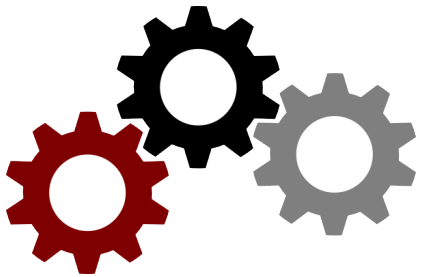
The Hierarchy

User

iRODS rules

Sys-admin/
iRODS admin

iRODS server rule base



Rule engine(s)

Micro services

Write your own data archiving policy/rule