# Introduction à iRODS

Bruno Bzeznik

Ingénieur de Recherche à GRICAD

ANF UST4HPC 2023 - Aussois

# iRODS

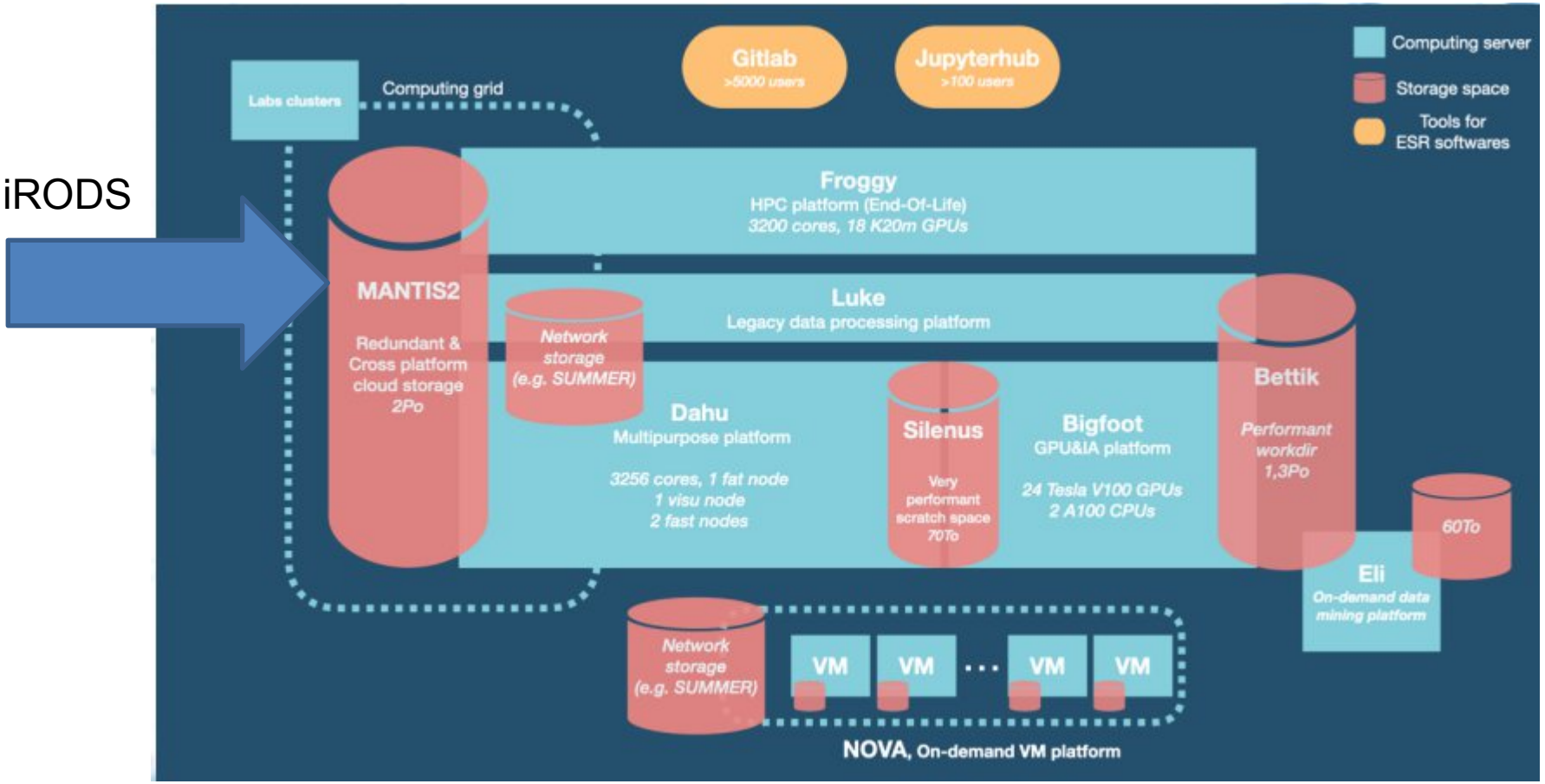


- Système de stockage distribué de type « objet » ou « grille de stockage » : l'unité de stockage est le fichier
- iRODS n'est pas POSIX et ne gère pas les blocs
- iRODS fonctionne à un haut niveau, en tant que service qui tourne sur des serveurs ayant des volumes de stockage POSIX
- Il offre un espace de nommage unique au sein d'une « zone »
- Il offre une API rest, mais c'est accessoire et HTTP n'est pas son protocole : il a son propre protocole de transfert multithreadé

# iRODS



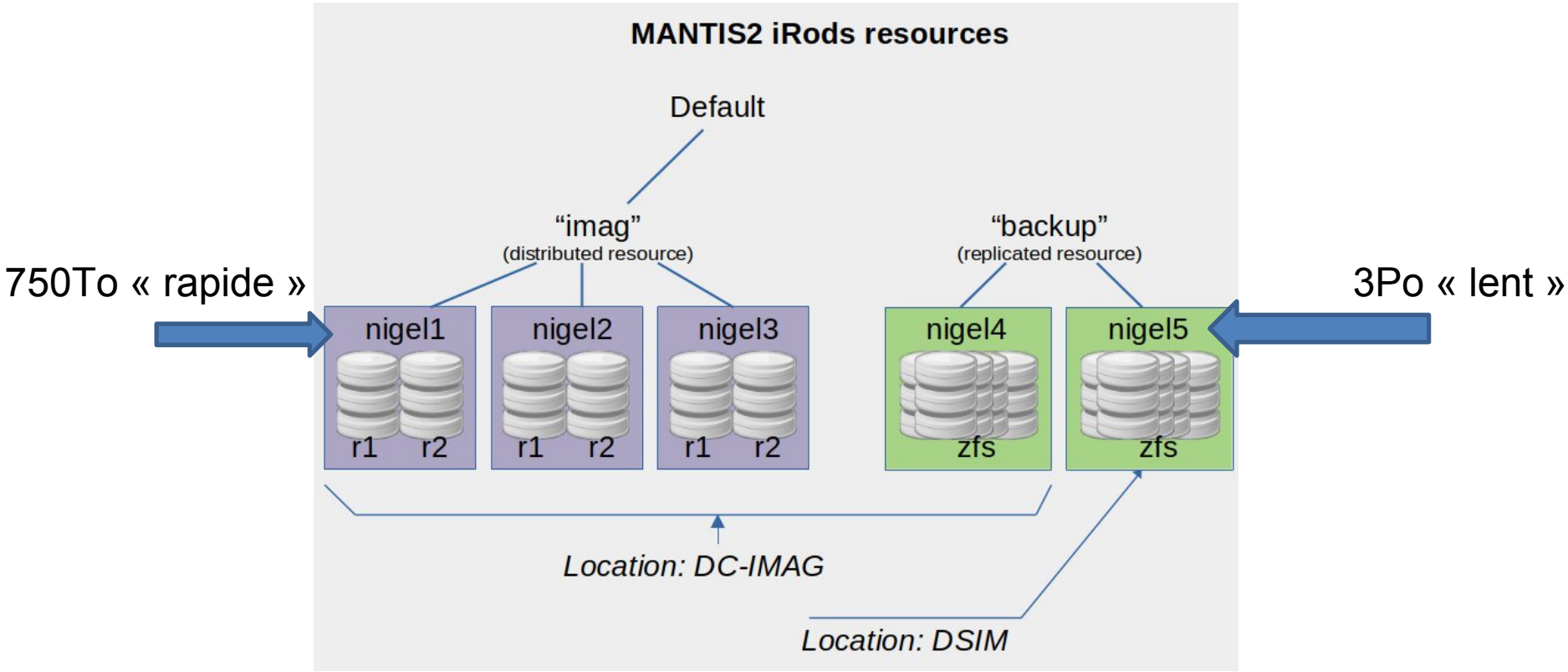
- Utilisé à GRICAD (CIMENT) depuis 2010, pour faciliter le stagein/stageout de fichiers dans des jobs « grille » mettant en jeu plusieurs clusters de calcul : « MANTIS »



# iRODS

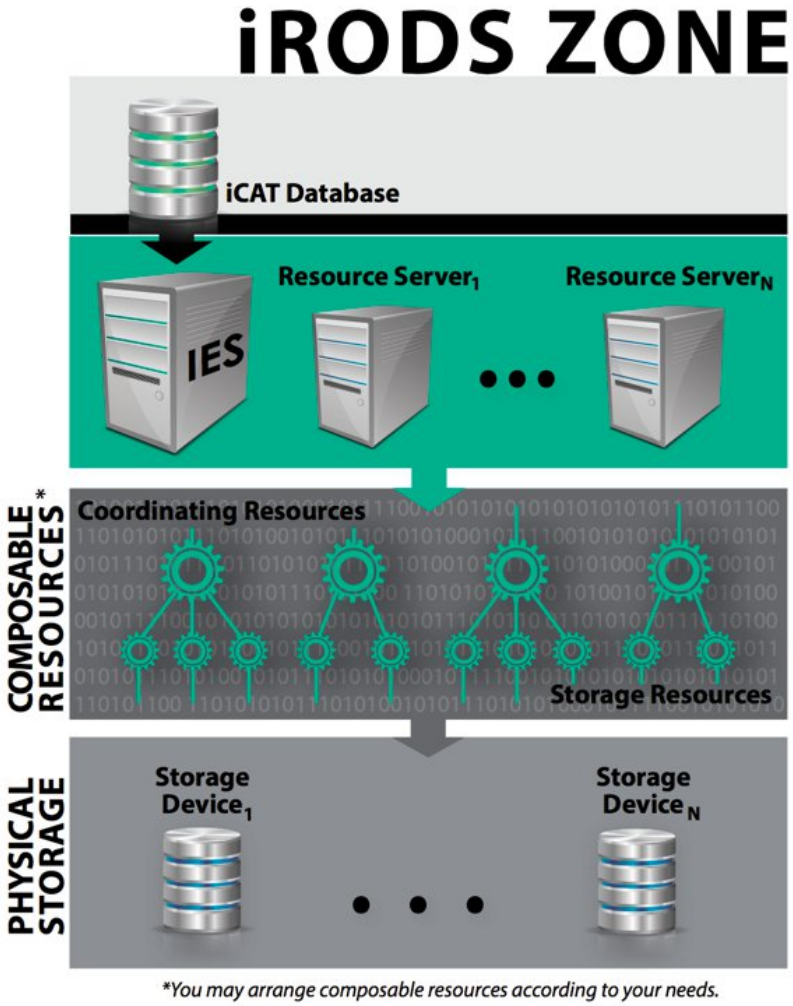


- iRods permet de gérer de l'hétérogénéité dans les ressources de stockage et sa conception permet une grande extensibilité de la capacité et des performances : scale-out



# Principes






iCat = Serveur iRods en mode « Provider » (un seul par Zone)

Resource Server = Serveur iRods en mode « Consumer »
(Un par serveur physique ; peut gérer plusieurs ressources )

Systèmes de fichiers POSIX (XFS, ZFS,...) ou autre ressource « objet » (S3, Ceph, HPSS,...)

# Coordinating Resources



- **RANDOM**
  Répartition aléatoire des fichiers sur les « storage resources » filles
- **LOAD BALANCING**
  Nécessite la mise en place du monitoring interne, et répartition en fonction de la charge
- **REPLICATION**
  Les fichiers sont répliqués sur toutes les ressources filles
- **PASSTHRU**
  Sert à mettre un poids en lecture et un poids en écriture permettant l'élection des ressources de stockage. Cette ressource ne prend forcément qu'une seule ressource de stockage fille

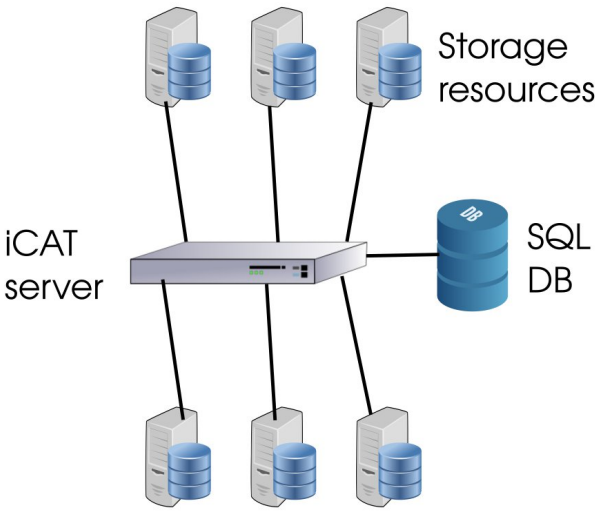
# Terminologie



**provider :** le serveur principal, hébergeant les meta-données. Anciennement appellé « iCat »
**consumer** : un serveur hébergeant les ressources de stockage et communiquant avec le provider et les clients. Appellé aussi « irodsServer ».
**zone** : un ensemble provider + consumers ; les zones peuvent être fédérées
**collection** : répertoire
**data_object** : fichier
**resource** : une zone de stockage physique, au sein d'un provider ; ou encore une « composition » de ressources (par exemple un regroupement de plusieurs ressources physiques)
**replicate** : copie d'un même fichier, sur des ressources différentes

# Workflow

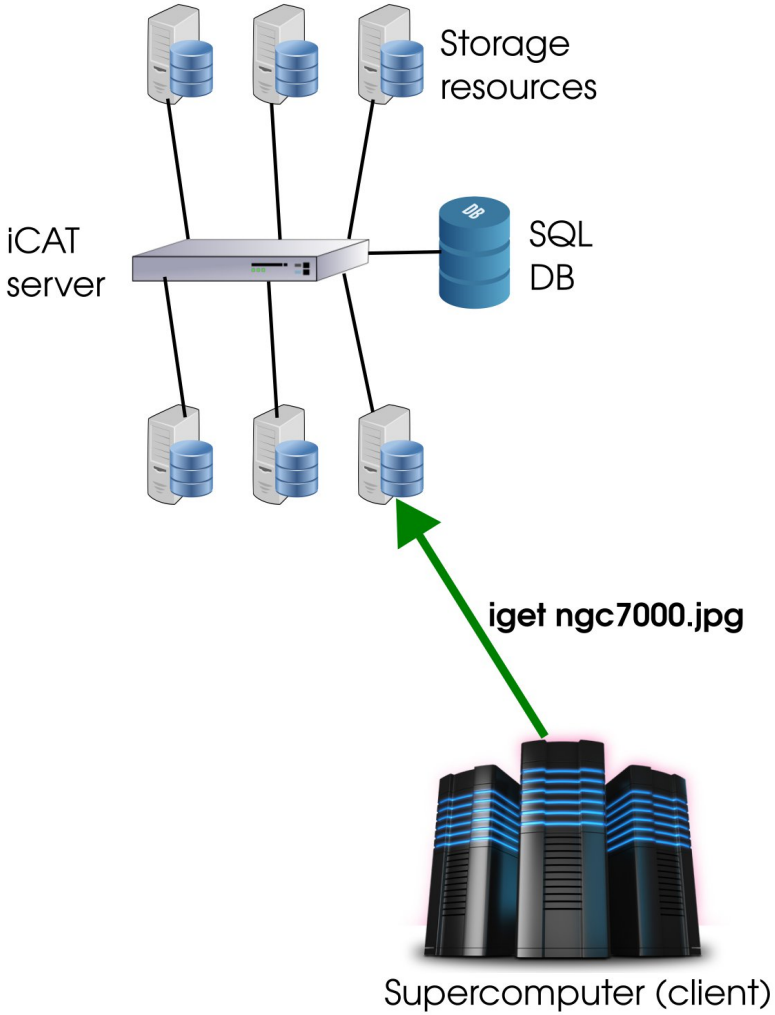





Le serveur iCat (provider) est un serveur de meta-données. Son backend est un serveur PostgreSQL

# Workflow



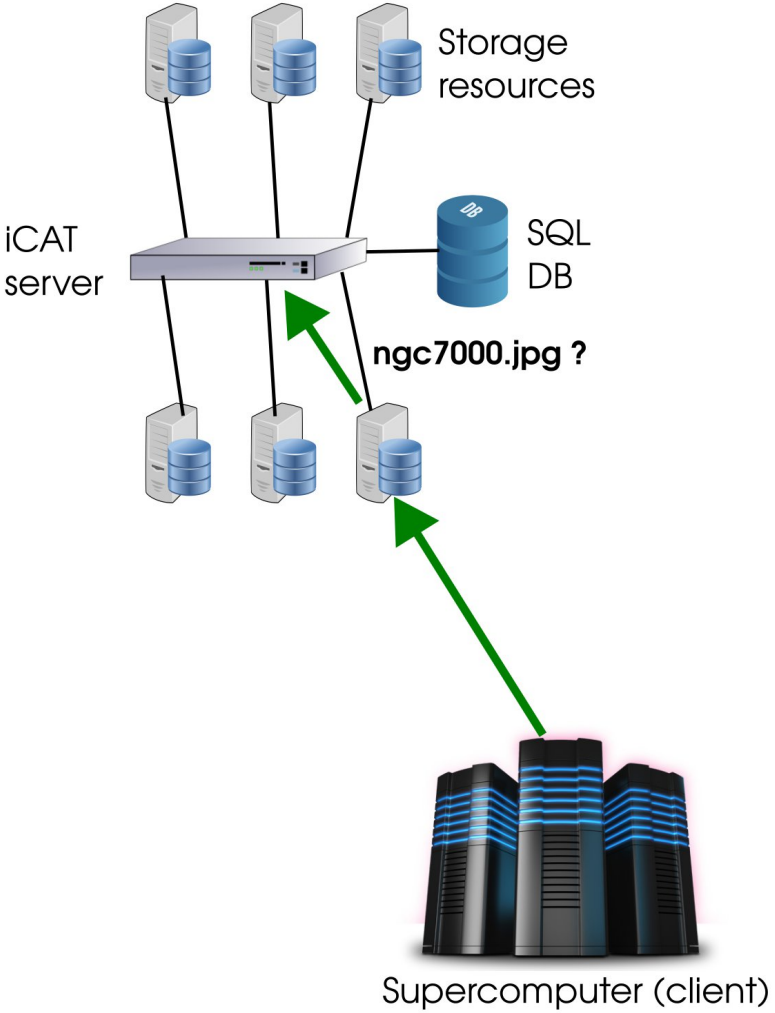



Les clients (ici, les nœuds d'un supercalculateur) sont connectés à un serveur de ressources (consumer) proche (au sens réseau).

Un client demande à récupérer un fichier...

# Workflow

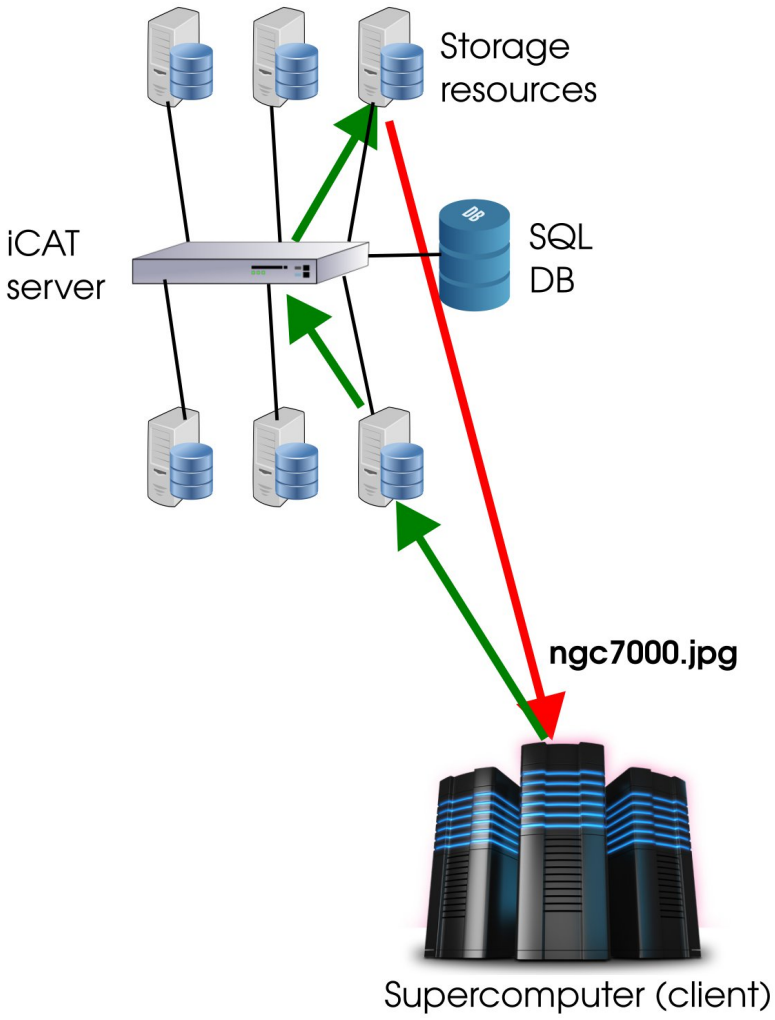





... le consumer transmet la requête au provider qui interroge la base de données des meta-data...

# Workflow






... une fois la ressource de stockage identifiée, un lien direct est établi entre le consumer et le client qui peut télécharger le fichier.

Ces opérations sont transparentes pour l'utilisateur.

Il se peut que le fichier soit répliqué sur plusieurs ressources. Le moteur de règles peut être configuré pour favoriser le choix d'un réplica.

# Clients : iCommands



Les « iCommands » constituent le client de base, en ligne de commande

- Déposer un fichier : **iput**

```
$ iput ngc7000.jpg
```

- Lister les fichiers : **ils**

```
$ ils
/ust4hpc/home/bzizou:
  changelog.gz
  ngc7000.jpg
$ ils -l
/ust4hpc/home/bzizou:
  bzizou            0 ust4hpc-data0-pt;ust4hpc-data0        71406 2023-06-21.10:18 & changelog.gz
  bzizou            0 ust4hpc-data1-pt;ust4hpc-data1       140685 2023-06-20.14:38 & ngc7000.jpg
$ ils -L
/ust4hpc/home/bzizou:
  bzizou            0 ust4hpc-data0-pt;ust4hpc-data0        71406 2023-06-21.10:18 & changelog.gz
        generic    /var/lib/irods/Vault//home/bzizou/changelog.gz
  bzizou            0 ust4hpc-data1-pt;ust4hpc-data1       140685 2023-06-20.14:38 & ngc7000.jpg
        generic    /var/lib/irods/Vault//home/bzizou/ngc7000.jpg
```

- Downloader un fichier : **iget**

```
$ iget ngc7000.jpg
```

# Clients : iCommands



## Commande d'admin : **iadmin**

```
$ iadmin help
Usage: iadmin [-hvV] [command]
A blank execute line invokes the interactive mode, where it
prompts and executes commands until 'quit' or 'q' is entered.
Single or double quotes can be used to enter items with blanks.
Commands are:
 lu [name[#Zone]] (list user info; details if name entered)
 lua [name[#Zone]] (list user authentication (GSI/Kerberos Names, if any))
 luan Name (list users associated with auth name (GSI/Kerberos)
 lt [name] [subname] (list token info)
 lr [name] (list resource info)
 ls [logical_path <string>|data_id <int>] [replica_number <int>|resource_hierarchy <string>] (list replica i
 lz [name] (list zone info)
 lg [name] (list group info (user member list))
 lgd name  (list group details)
 mkuser Name[#Zone] Type (make user)
 moduser Name[#Zone] [ type | comment | info | password ] newValue
 aua Name[#Zone] Auth-Name (add user authentication-name (GSI/Kerberos)
 rua Name[#Zone] Auth-Name (remove user authentication name (GSI/Kerberos)
 rpp Name  (remove PAM-derived Password for user Name)
 rmuser Name[#Zone] (remove user, where userName: name[@department][#zone])
 rmdir Name (remove directory)
 mkresc Name Type [Host:Path] [ContextString] (make Resource)
 modresc Name [name, type, host, path, status, comment, info, free_space, context, rebalance] Value (mod Res

[..]
```

# Clients : API



C++, Java, Go.... mais aussi python :

python-irodsclient : https://github.com/irods/python-irodsclient

```
from irods.session import iRODSSession
[...]
for obj in irods_session.collections.get(coll).subcollections:
     image = irods_session.data_objects.get(obj.path)
```

# Meta-données



Les meta-données peuvent être enrichies à volonté par les utilisateurs. C'est un des grands atouts d'iRods car on peut ensuite sélectionner des fichiers indépendemment de l'arborescence classique.

```
$ imeta add -d ngc7000.jpg temps_de_pose 3 heures
$ imeta ls -d ngc7000.jpg
AVUs defined for dataObj /ust4hpc/home/bzizou/ngc7000.jpg:
attribute: temps_de_pose
value: 3
units: heures
$ imeta qu -d temps_de_pose = 3
collection: /ust4hpc/home/bzizou
dataObj: ngc7000.jpg
```

L'exploitation des meta-données peut-être très pratique lorsque l'on utilise une API, par exemple le client python : https://github.com/irods/python-irodsclient#working-with-metadata

# Règles



- Les règles sont le « R » de i**R**ODS
- Les règles les plus courantes sont dans le fichier **/etc/irods/core.re** et permettent de configurer le comportement du système à chaque point d'entrée : lorsqu'un fichier est créé, lorsqu'un fichier est lu, répliqué, etc...
- On peut écrire ses propres règles. Exemple, pour une réplication automatique sur une ressource donnée :

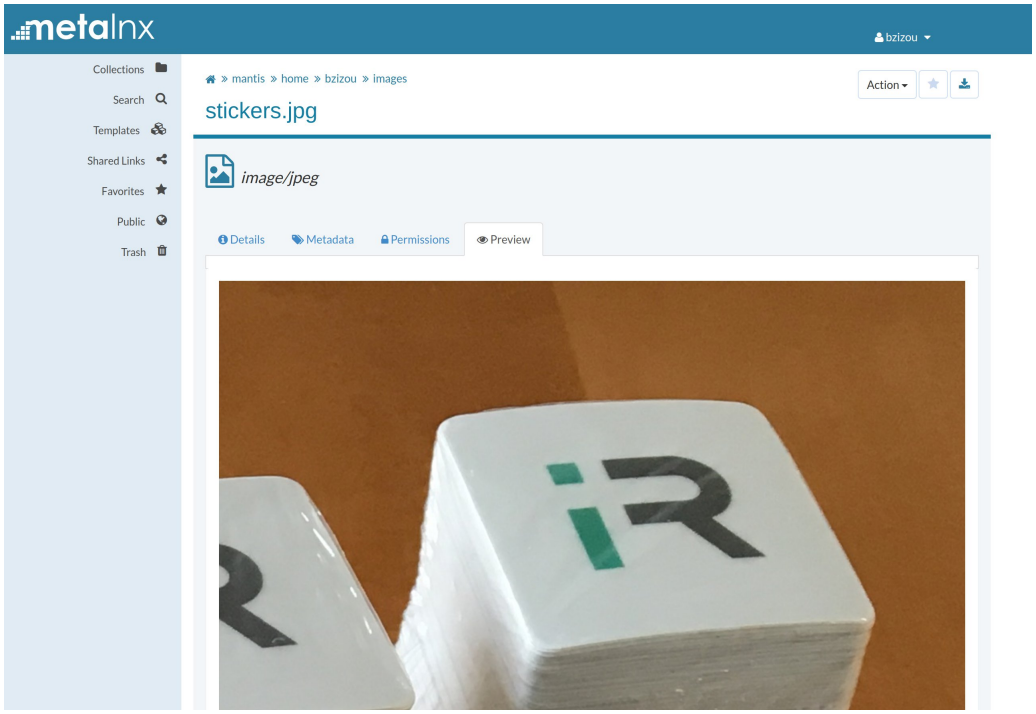
```
pep_api_data_obj_get_pre(*INSTANCE_NAME, *COMM, *DATAOBJINP, *BUFFER, *PORTAL_OPR_OUT)
{
  msiDataObjRepl(*DATAOBJINP.obj_path,"ust4hpc-data1-pt",*status);
}
```

- Des plugins permettent d'activer différents moteurs de règles, en particulier le moteur python
  -> https://github.com/bzizou/sysadmin/blob/master/irods/PYTHON_RULE_ENGINE.md

- Voir la doc des « Policy Enforcement Points » :
  https://docs.irods.org/4.3.0/plugins/dynamic_policy_enforcement_points/

# Clients : Web






*Installation conseillée sous forme de containers*

## Interface web : metalnx

## Protocole Webdav : Davrods

```
[bzizou@bart:~]$ cadaver https://mantis.univ-grenoble-alpes.fr/davrods/

Username: bzizou

Password:
dav:/davrods/> ls

Listing collection `/davrods/': succeeded.

Coll:   povray_results_mantis2            0  Jun  3  2020

Coll:   system                            0  Jun 30 16:48

        10GBtestfile              10485760000  Nov 13 14:26

        test1                       161886424  Aug 26 15:08

dav:/davrods/> get test1
```

# Meta-données



Les meta-données peuvent être enrichies à volonté par les utilisateurs. C'est un des grands atouts d'iRods car on peut ensuite sélectionner des fichiers indépendemment de l'arborescence classique.

```
$ imeta add -d ngc7000.jpg temps_de_pose 3 heures
$ imeta ls -d ngc7000.jpg
AVUs defined for dataObj /ust4hpc/home/bzizou/ngc7000.jpg:
attribute: temps_de_pose
value: 3
units: heures
$ imeta qu -d temps_de_pose = 3
collection: /ust4hpc/home/bzizou
dataObj: ngc7000.jpg
```

L'exploitation des meta-données peut-être très pratique lorsque l'on utilise une API, par exemple le client python : https://github.com/irods/python-irodsclient#working-with-metadata

# Règles



- Les règles sont le « R » de i**R**ODS
- Les règles les plus courantes sont dans le fichier **/etc/irods/core.re** et permettent de configurer le comportement du système à chaque point d'entrée : lorsqu'un fichier est créé, lorsqu'un fichier est lu, répliqué, etc...
- On peut écrire ses propres règles. Exemple, pour une réplication automatique sur une ressource donnée :

```
pep_api_data_obj_get_pre(*INSTANCE_NAME, *COMM, *DATAOBJINP, *BUFFER, *PORTAL_OPR_OUT) {
  msiDataObjRepl(*DATAOBJINP.obj_path,"ust4hpc-data1-pt",*status);
}
```

- Des plugins permettent d'activer différents moteurs de règles, en particulier le moteur python
  -> https://github.com/bzizou/sysadmin/blob/master/irods/PYTHON_RULE_ENGINE.md

# Quotas



- **Gérer les quotas : `iadmin suq`**
- **Afficher les quotas : `iquota`**

A lancer régulièrement en CRON :

```
/usr/bin/iadmin cu
```

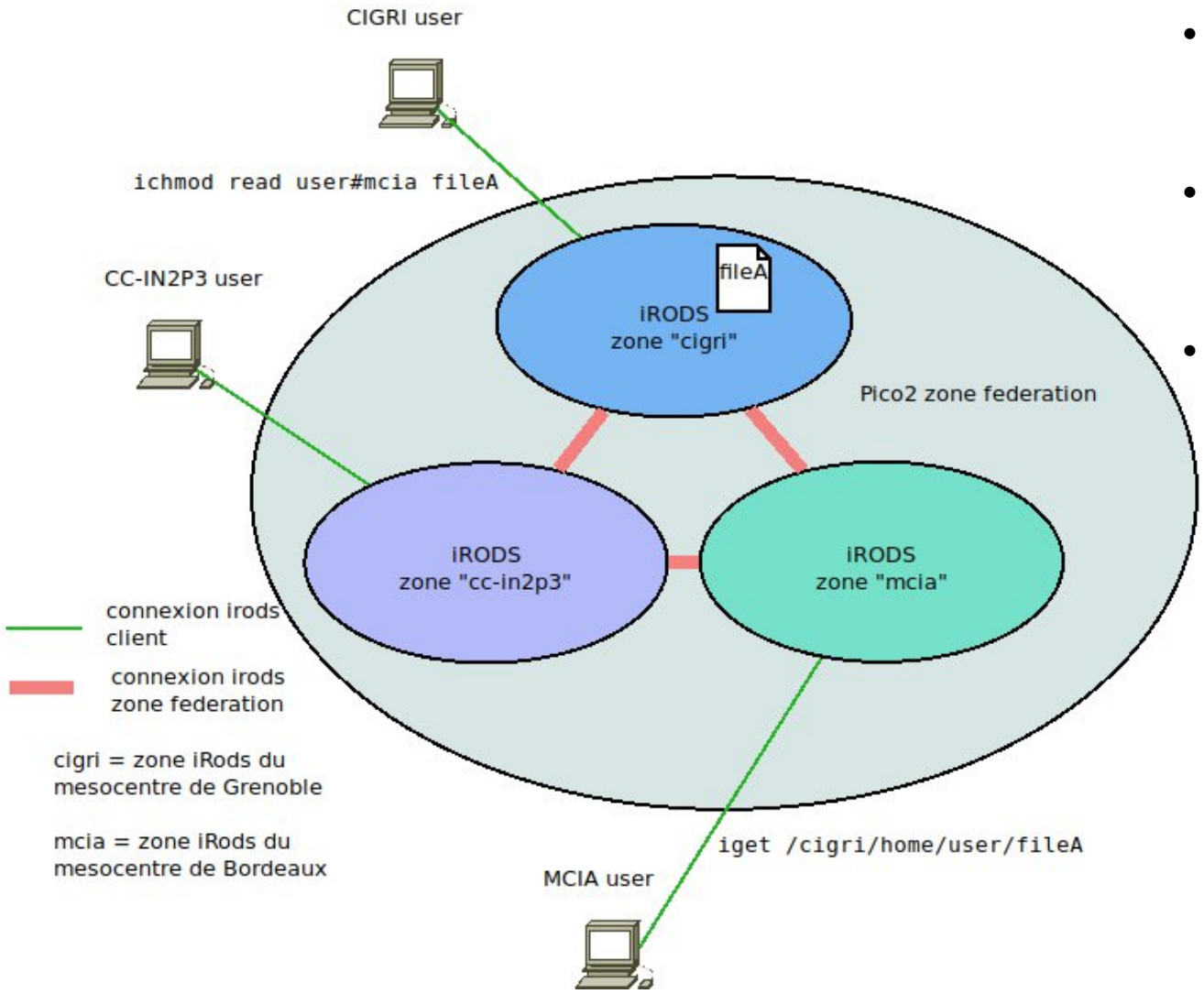
# Authentification



- Interne (users/password dans l'iCat)
- PAM et par conséquent LDAP (c'est ce que nous utilisons à Gricad). Nécessite tout de même une synchro avec le système interne (les utilisateurs doivent être déclarés)
- OpenID

# Fédération de zones

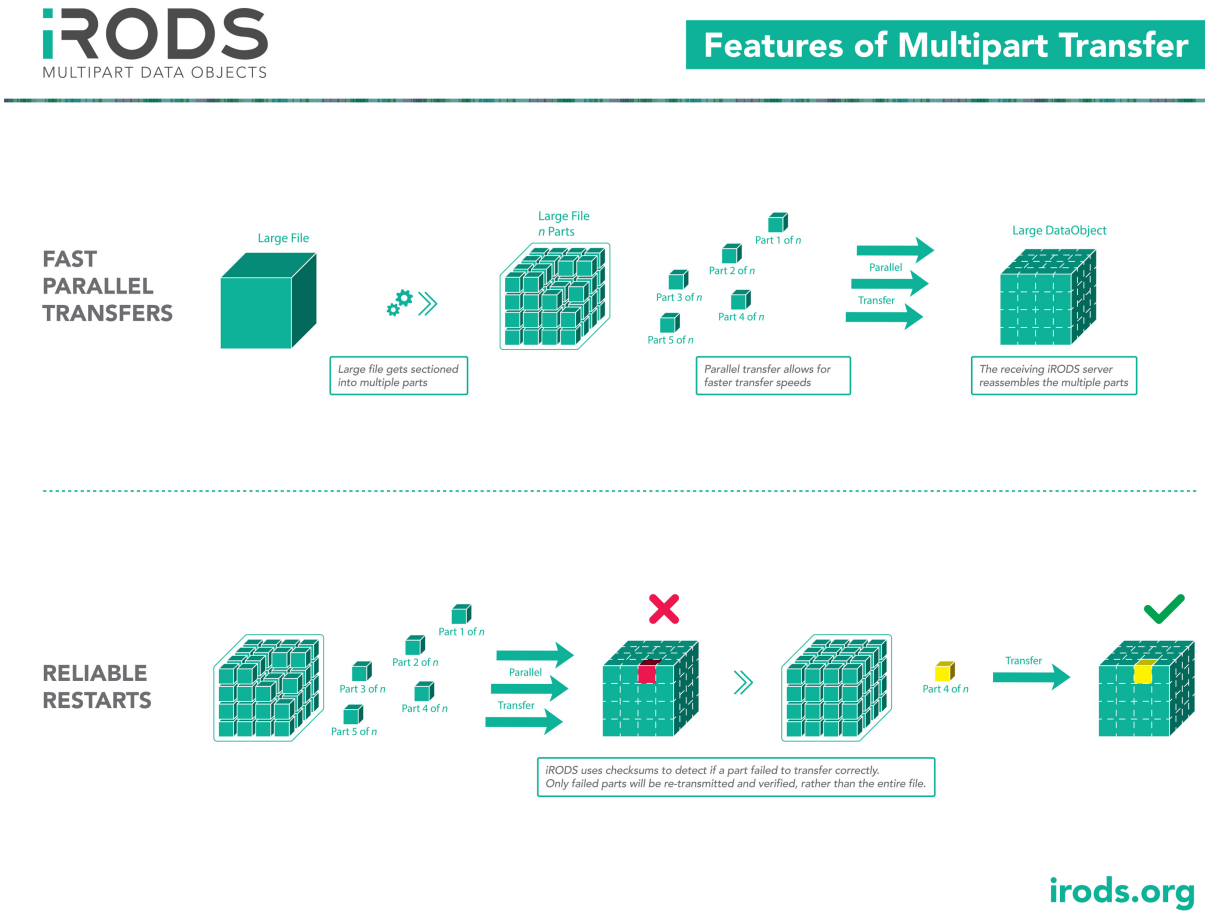





- Le nom de la zone correspond à la racine de l'arborescence des fichiers
- Chaque zone peut avoir son propre système d'authentification
- Les utilisateurs d'une zone octroient des droits d'accès sur leurs fichiers, aux utilisateurs de la zone voisine

# Au sujet des transfert de fichiers






Petits fichiers vs gros fichiers (>32Mo par défaut) : les petits fichiers passent par la ressource cliente. Pour les gros fichiers, un canal de communication multi-socket est créé entre le client et la ressource serveur -> Attention au filtrage (utilise de nombreux ports)

# Refs



- **https://docs.irods.org/**

- https://indico.in2p3.fr/event/23075/contributions/90180/attachments/61890/84544/08_Moteur_Regles.pdf
- https://slides.com/jasoncoposky/cines-2020-rule-engine-plugins
- https://github.com/irods/irods_training/blob/main/beginner/irods_beginner_training_2019.pdf
- https://github.com/irods/irods_rule_engine_plugin_python/blob/main/README.md

# TP



Sujet : https://github.com/bzizou/sysadmin/blob/master/irods/TP_sujet.md