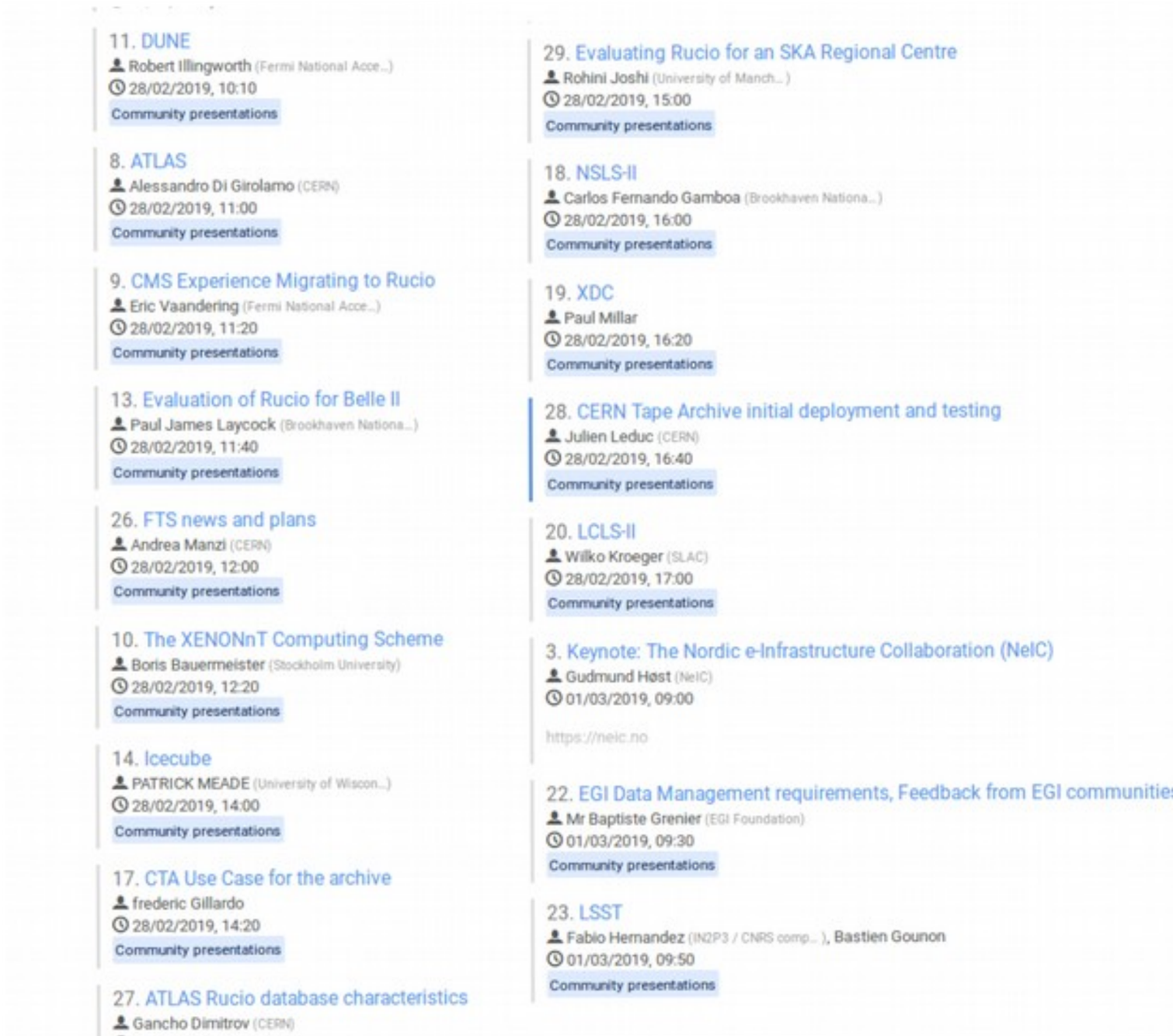
Rucio Tutorial

- NTF004F2019
- Florido Paganelli
- florido.paganelli@hep.lu.se

Why Rucio

- Very popular among CERN experiments, getting popular also for others
- Uses modern technologies
- It’s open source
- Intensively developed

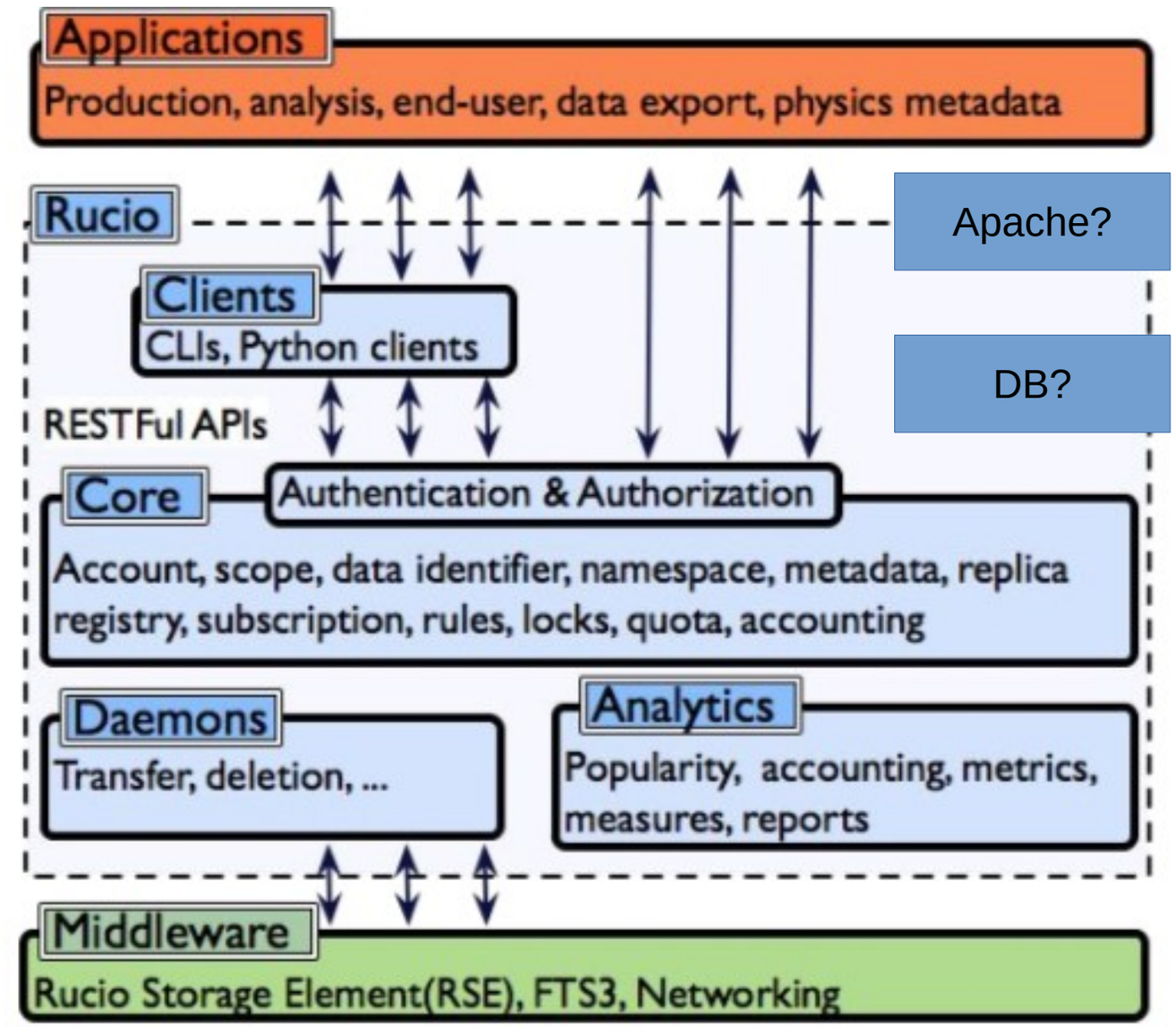


What does it do

- Data management services
  - Creation of collections of data
  - Definition of datasets
  - Replication
  - Metadata management
  - Usage Logging
  - Access to **existing** storage elements
    - Experiments need to **already have some storage** with an interface Rucio can interact with
    - Download/upload
  - Definition of access control rules/systems
    - It hooks to pre-existing infrastructures (PKI, Tokens…)

Architecture

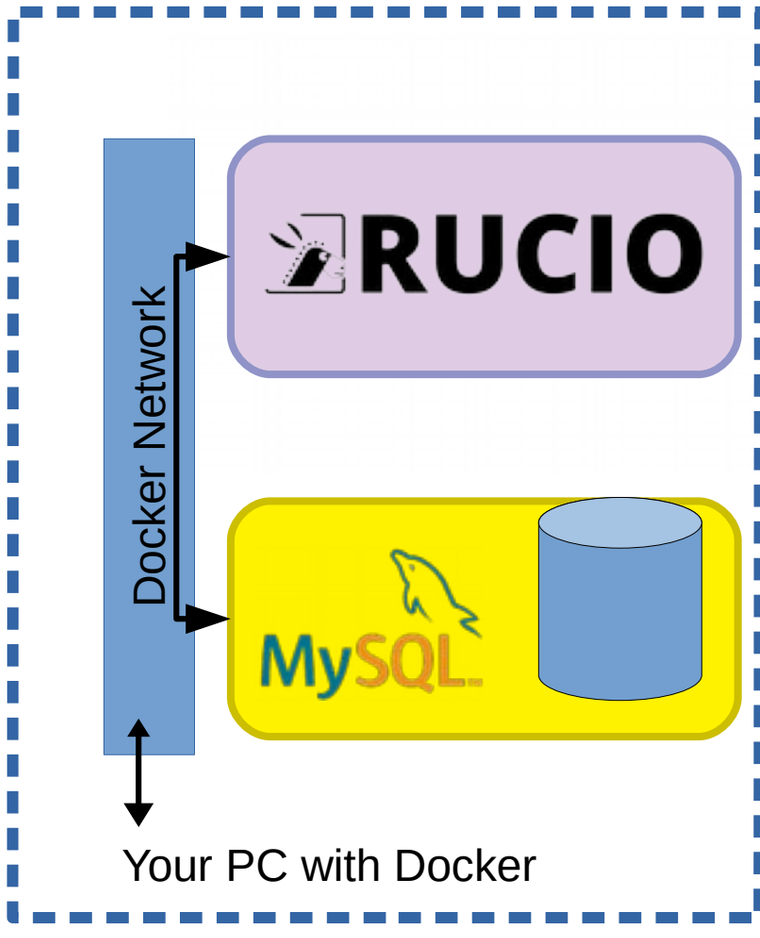
- Open source
- Documentation is not very clear :(
- Python CLI/API clients
- Collection of python scripts and daemons
- Undocumented:
  - Apache web server
  - Data is stored in a database – can be hooked to different DBs.



- Deployment:
  - Distributed as a PiPY package, but suggested deployment is as a docker container…
  - Most support and deployments seems to be done ad-hoc for each experiment by the developers themselves.

The demo

- Instructions: https://rucio.readthedocs.io/en/latest/rucio_demo.html
- Showcases Rucio features, but does not really clarify what they are for. I'll try to do during this lecture.
- To install it, one needs
  - A copy of rucio's git code
  - docker-composer
- The composer installs and starts
  - A rucio container based on Linux CentOS7
    - We will login into this one
  - A mysql database docker container
  - **The demo is a typical docker app.**



The demo – installation

- There is a **bug** in the official repository that prevents the demo to work. Luckily this is open source hence I could fix the bug.
- Find a spot in your disk where the repository will be created. A folder named **`rucio`** will be created after this operation, where we will run most docker commands.
- Fetch the working version from my GIT fork:

```
git clone https://github.com/floridop/rucio.git
cd rucio
git checkout emergencyfix
```

- Run docker-compose

```
sudo docker-compose --file etc/docker/demo/docker-compose.yml up -d
```

- You can now proceed with the installation instructions in the official docker page “**Checking the Containers**” https://hub.docker.com/r/rucio/demo
- And we can go back to the readthedocs page mentioned in the previous slide.

Docker-compose.yml

```
# Authors:
# - Thomas Beermann, <thomas.beermann@cern.ch>, 2018
Version: "2"
services:
  rucio:
    build:
      context: .
      args:
        - http_proxy
        - https_proxy
        - no_proxy
    ports:
      - "443:443"
    links:
      - mysql:mysql
    depends_on:
      - "mysql"
    command: ["httpd", "-D", "FOREGROUND"]

Mysql:
  image: mysql/mysql-server:5.7
  environment:
    - MYSQL_RANDOM_ROOT_PASSWORD=True
    - MYSQL_DATABASE=rucio
    - MYSQL_USER=rucio
    - MYSQL_PASSWORD=rucio
    - MYSQL_ROOT_HOST=%
```

Containers

network

Docker-compose.yml

```
# Authors:
# - Thomas Beermann, <thomas.beermann@cern.ch>, 2018
Version: "2"
services:
  rucio:
    build:
      context: .
      args:
        - http_proxy
        - https_proxy
        - no_proxy
    ports:
      - "443:443"
    links:
      - mysql:mysql
    depends_on:
      - "mysql"
    command: ["httpd", "-D", "FOREGROUND"]
  Mysql:
    image: mysql/mysql-server:5.7
    environment:
      - MYSQL_RANDOM_ROOT_PASSWORD=True
      - MYSQL_DATABASE=rucio
      - MYSQL_USER=rucio
      - MYSQL_PASSWORD=rucio
      - MYSQL_ROOT_HOST=%
```

Use files and Dockerfile in the current folder

network

Start an apache webserver

Containers

Mysql configuration using env variables

Dockerfile

```
FROM rucio/rucio-systemd-cc7

ADD ca.repo /etc/yum.repos.d/ca.repo
RUN pip install --upgrade pip
RUN pip install --upgrade setuptools
RUN pip install --ignore-installed ipaddress
RUN pip install M2Crypto==0.32
RUN pip install rucio rucio-webui

WORKDIR /opt
```

Installs rucio

```
ADD rucio.cfg /opt/rucio/etc/
RUN chmod 644 /opt/rucio/etc/rucio.cfg
ADD alembic.ini /opt/rucio/etc/
RUN chmod 644 /opt/rucio/etc/alembic.ini
ADD aliases-py27.conf /opt/rucio/etc/web/
RUN chmod 644 /opt/rucio/etc/web/aliases-py27.conf
ADD ui-aliases-py27.conf /opt/rucio/etc/web/
RUN chmod 644 /opt/rucio/etc/web/ui-aliases-py27.conf
ADD automatix.json /opt/rucio/etc/

RUN mkdir /opt/rucio/tools
ADD dump_schema.py /opt/rucio/tools
RUN chmod 755 /opt/rucio/tools/dump_schema.py
ADD activate_rucio_global_completion.sh
/opt/rucio/tools
RUN cat
/opt/rucio/tools/activate_rucio_global_completion.sh
>> /root/.bashrc
```

Add rucio demo config files to image

Dockerfile

```
RUN mkdir /var/log/rucio
RUN mkdir /var/log/rucio/trace
RUN chmod 777 /var/log/rucio

ADD httpd.conf /etc/httpd/conf/httpd.conf
ADD rucio.conf /etc/httpd/conf.d/rucio.conf

ADD certs/ca.pem /opt/rucio/etc/web/CERN-bundle.pem
ADD certs/ca.pem /opt/rucio/etc/web/ca.crt
ADD certs/usercert.pem /opt/rucio/etc/web/usercert.pem

ADD certs/server.crt /etc/grid-security/hostcert.pem
ADD certs/server.key /etc/grid-security/hostkey.pem
RUN chmod 400 /etc/grid-security/hostkey.pem

ADD setup_demo.sh /
ADD setup_data.py /
ADD wait-for-it.sh /

WORKDIR /opt/rucio
```

Add rucio demo webserver configuration files to image

```
RUN rm /etc/httpd/conf.d/ssl.conf
/etc/httpd/conf.d/autoindex.conf
/etc/httpd/conf.d/userdir.conf
/etc/httpd/conf.d/welcome.conf
```

Remove default webserver config

```
EXPOSE 443
```

network

```
ENV PATH $PATH:/opt/rucio/bin
```

Rucio CLI

The running containers

- `demo_rucio` - Runs the rucio server and daemons, contains the clients. We will use if for the tutorial.
- `mysql/mysql-server:<tag>` - Used by Rucio to store any kind of information. We will not use this machine directly but through the rucio commands.

The scripts

- `setup_demo.sh`
  - Generates two datasets with two files each, named `AOD.<somehash>` , using `setup_data.py`
  - Creates two rucio-managed Storage Elements on a local filesystem:
    `/tmp/SITE1_DISK`
    `/tmp/SITE2_DISK`
  - Creates scopes and Datasets and adds the files to them.
  - Creates two rucio users `root` and `jdoe`
  - All the above is done through setup_data.py
- => Some sections “Configuring Rucio” in the demo documentation can be skipped.

Quick terminology recap

- **FILE**: well, you know.
- **DATASET**: a collection of files
- **CONTAINER**: a collection of datasets
- **DID**: full name for a a file, a dataset or a container, in the form SCOPE:NAME
  - Same as Logical File Name LFN
- **SCOPE**: beginning of each DID name, defines an authoritative area. Users have their own scope.
- **RSE**: Rucio Storage Element, a logical representation of some physical storage.
- **Meta-data attributes**: strings that describe objects
- **REPLICA**: a managed copy of a file

Login into the rucio container

- `sudo docker exec -it demo_rucio_1 /bin/bash`
- `rucio -h` shows a list of commands. TAB autocomplete will work, but sometimes will also just give you the content of the folder you are in.
- `rucio <command> -h` shows a list of options for a given command. TAB autocomplete will work, but sometimes will also just give you the content of the folder you are in.

Rucio basics

Execute the following commands and check their output.

- `rucio whoami`
  - Shows info about your current user.
- `rucio list-rses`
  - Shows info about existing Rucio Storage Elements.
- `rucio list-rse-attributes <RSE>`
  - Shows RSE attributes. These can be used to help the researcher identify where certain data is stored.
- `rucio list-rse-usage <RSE>`
  - Shows info about the space used in the RSE (but not the available space!!!)

Rucio scopes, datasets and files

- `rucio list-datasets-rse <RSE>`
  - Shows the available datasets on a given RSE
- `rucio list-scopes`
  - Shows info about existing scopes, which are labels composing the beginning of a name of a dataset or file. They can be used to identify the kind of data you're working on, or the subject who generated it. Usually each user has their own scope.
- `rucio list-dids <DID-espression>`
  - A DID-expression is always of the form SCOPE:NAME. Rucio accepts wildcards, but it's not that versatile. In this tutorial we can use tests:* as an example.

Rucio scopes, datasets and files

- `rucio list-files <DID>`
  - Shows the files in a given dataset.
- `rucio get-metadata <DID> [<DID>]..`
  - Shows metadata of the specified DID.
- `rucio download <DID>`
  - Downloads a did, either an entire dataset or a single file inside it. Try! Browse the downloaded files/datasets.

Rucio managing files

- Let’s create some file to upload in the tests scope.

  `echo “This is my data file” > myowndata.txt`

- Let’s upload it as a file in the tests scope, in the first storage element.

  `rucio upload --scope tests --rse SITE1_DISK  myowndata.txt`

- `Where is the file?`

  `rucio list-dids tests:*`

  `rucio list-dids tests:myownfile.txt`

  `…?????`

- `rucio list-rules --account=root`

  definitely weird.

Rucio managing datasets

- Let’s add our file to an existing dataset
  `rucio attach <destinationdataset> tests:myowndata.txt`
- Let’s check that it is now in the list of files:
  `rucio list-files <destinationdataset>`
- Let’s download the dataset again...
  `rucio download <destinationdataset>`
  it’s there!
- Run script that should be automated.
  `/usr/bin/rucio-judge-evaluator --run-once`
- Now the number of files is consistent.
  `rucio list-rules --account=root`

Rucio creating datasets

- Let's add our file to an existing dataset

  `rucio add tests:myowndataset`

- Let's check that it is now in the scope:

  `rucio list-dids tests:*`

- Should be empty:

  `rucio list-files tests:myowndataset`

- Let's remove the file from the other dataset and put it in the newly created

  `rucio detach <olddataset> tests:myowndata.txt`
  `rucio attach tests:myowndataset tests:myowndata.txt`
  `rucio list-files tests:myowndataset`

- Now the number of files is not consistent...

  `rucio list-rules --account=root`

- Run script to update the database again.

  `/usr/bin/rucio-judge-evaluator --run-once`

Rucio adding metadata

- Adding metadata to a dataset:

```
rucio add-did-meta --did tests:myowndataset \
--key securitylevel --value topsecret
```

- Retrieve metadata information:

```
rucio get-did-meta tests:myowndataset
```

- Find datasets with a given metadata information:

```
rucio list-dids-by-meta --scope tests securitylevel=topsecret
```

- Remove metadata:

```
rucio delete-did-meta --did tests:myowndataset \
--key securitylevel
```

Rucio replication

- We will instruct rucio to copy our dataset over the second RSE. This is done via a *rule*:

```
rucio add-rule tests:myowndataset 1 SITE2_DISK
```

- Now the number of files is not consistent...

```
rucio list-rules --account=root
```

- Run script to update the database again.

```
/usr/bin/rucio-judge-evaluator --run-once
```

- The actual replication happens via one of the daemons, but the documentation doesn’t clarify...

Rucio administration

- Check user accounts

```
rucio-admin account list
```

- Check user identities (ways to authenticate) and attributes (info)

```
rucio-admin account list-identities <accountname>
rucio-admin account list-attributes <accountname>
```

- Manage scopes

```
rucio-admin scopes add --account=root --scope myownscope
rucio-admin scopes list
```

- Add attributes to RSE:

```
rucio-admin rse set-attribute --rse SITE2_DISK \
  --key defaultusage --value replicaonly
```

- Check RSE info:

```
rucio-admin rse info SITE2_DISK
```

- List RSE by attribute:

```
rucio list-rses --expression 'defaultusage=replicaonly'
```

LUND
UNIVERSITY

References

- Rucio documentation https://rucio.readthedocs.io/
-