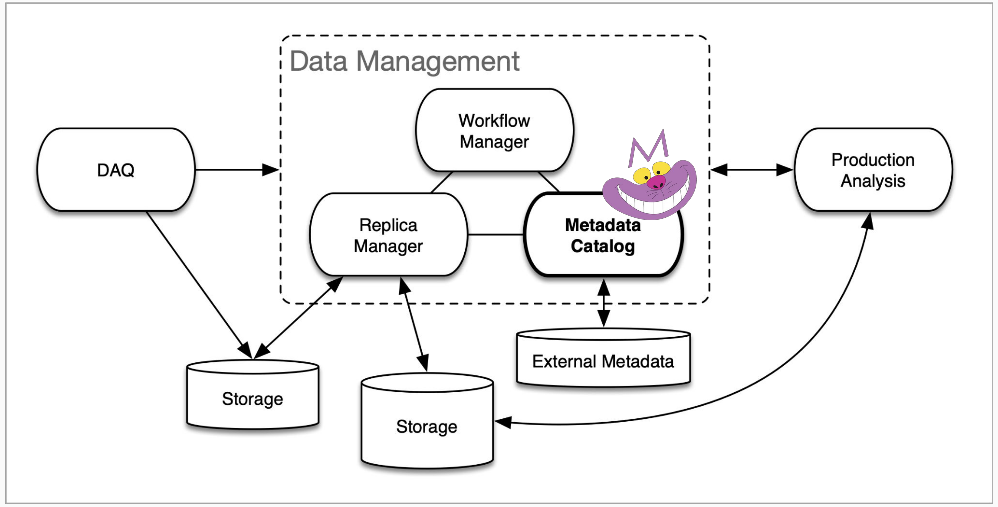
# MetaCat and Tokens

Igor Mandrichenko
FIFE meeting
10/14/2021

## Goals for this presentation

- To present another use case for the Tokens Task Force to consider
  - Web based application with client side components
- To see how well current MetaCat implementation and assumptions agree with the direction of the Tokens Task Force
- To see what needs to be adjusted and where
  - The earlier the better

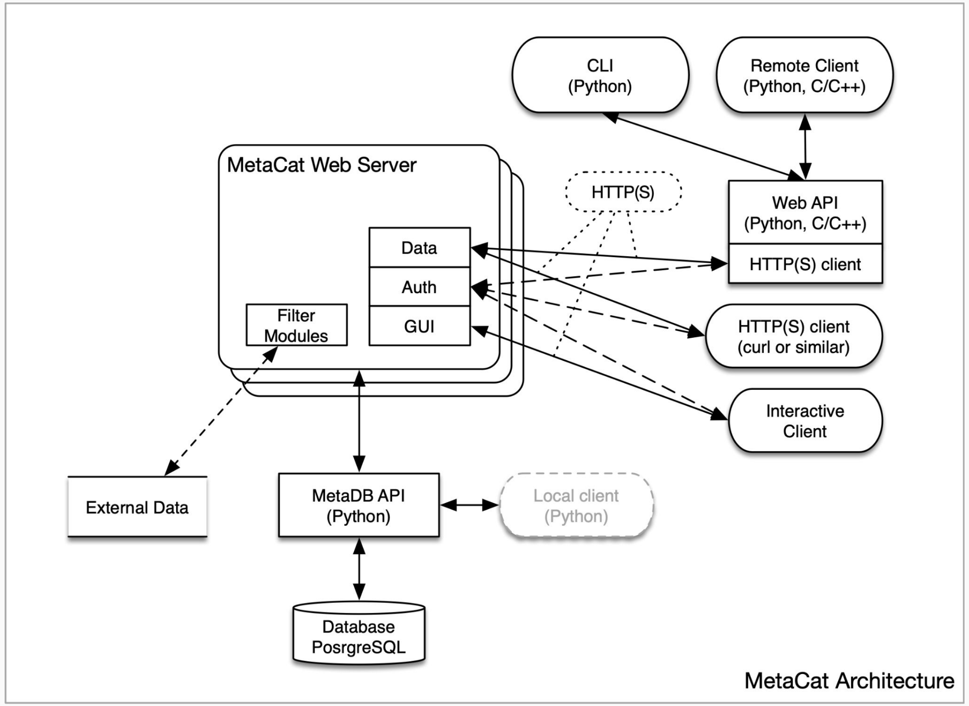
# What is MetaCat ?



MetaCat is a Metadata Catalog designed to be used in data management systems

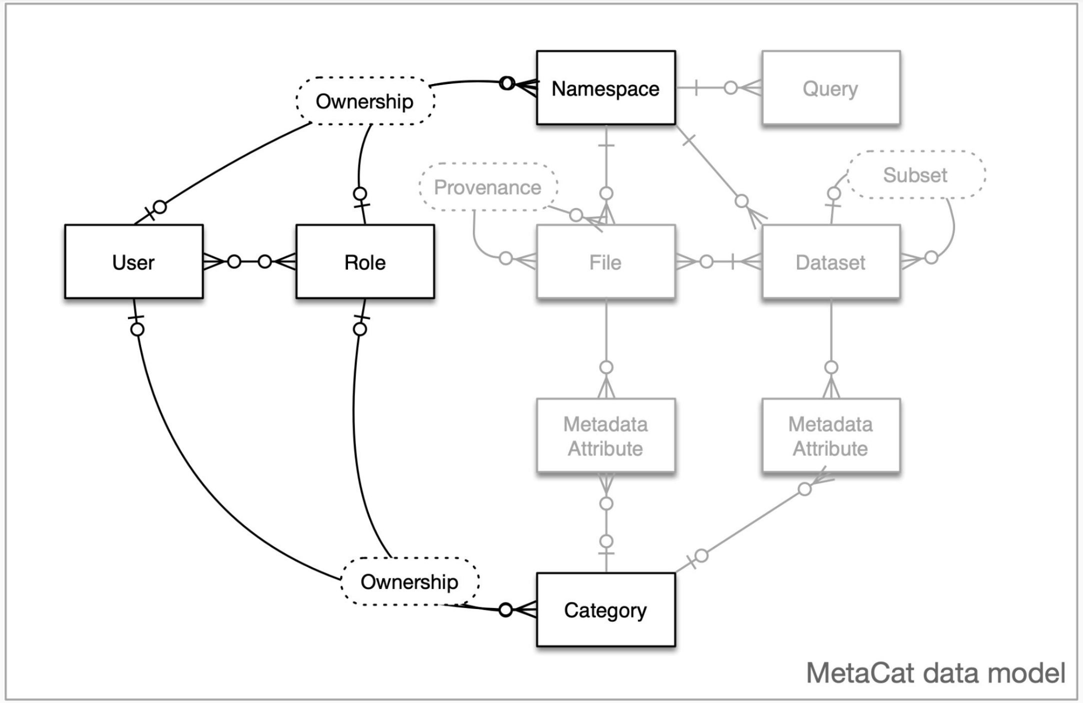
- Compatible but not dependent or integrated with Rucio
- Roughly same functionality as SAM metadata catalog
- Target experiment: DUNE
  - Not DUNE-centric

# MetaCat Architecture



- Web server application
  - W3C standard HTTP(s)
- Server side components
  - Authentication
  - Data (REST)
  - GUI
- Client side components
  - HTTP/REST client
  - Web API
  - CLI
  - Token management

# MetaCat authorization model

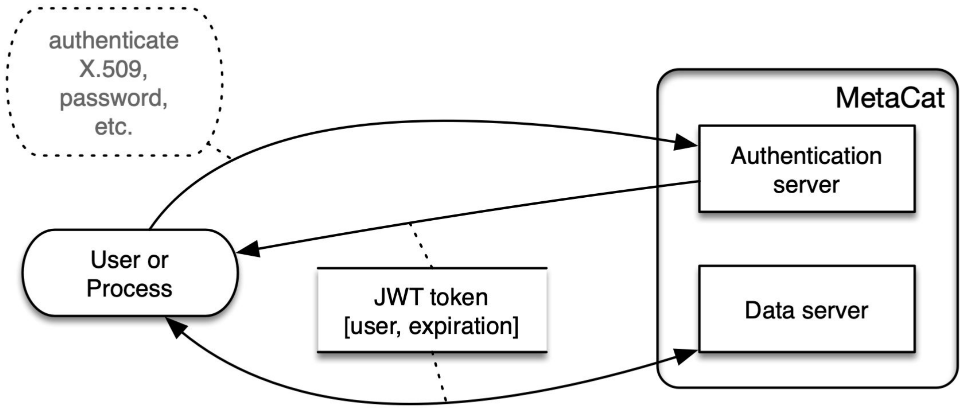


MetaCat data model

Mixed user/role-based authorization/ownership schema

- Namespace and Category can be owned by a user or a role
- User belongs to zero or more roles
- User identified by username
- User-role, ownership relationships are recorded in the MetaCat database

# Tokens and Client Authentication

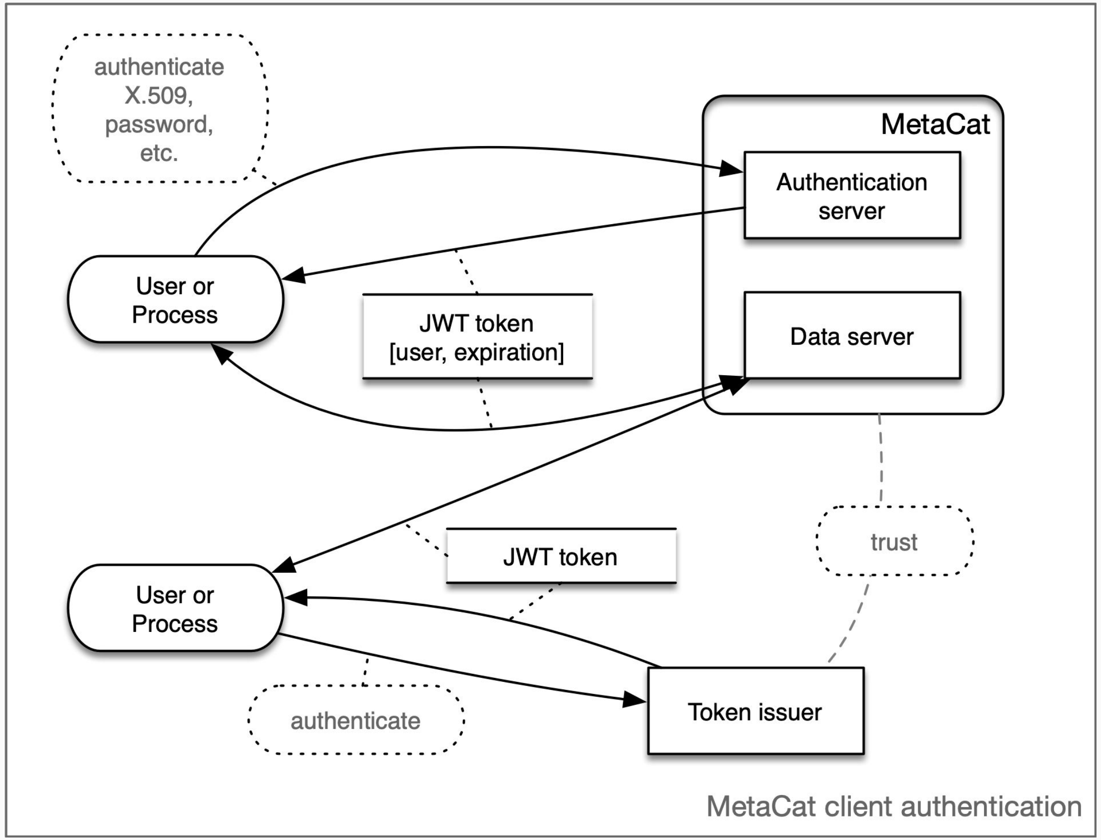

authenticate X.509, password, etc.
MetaCat
Authentication server
User or Process
JWT token [user, expiration]
Data server

1. Contact authentication server
2. Obtain token
3. Store token locally and/or in memory
4. Until the token expires:
    a. Present token to the data server
5. Go to (1)

MetaCat client authentication

- MetaCat auth server acts as token issuer
- Implemented authentication mechanisms
    - Password (LDAP, local hashed)
    - Digest RFC2617
    - X.509
- Token is signed with secret key shared between auth, data and GUI servers

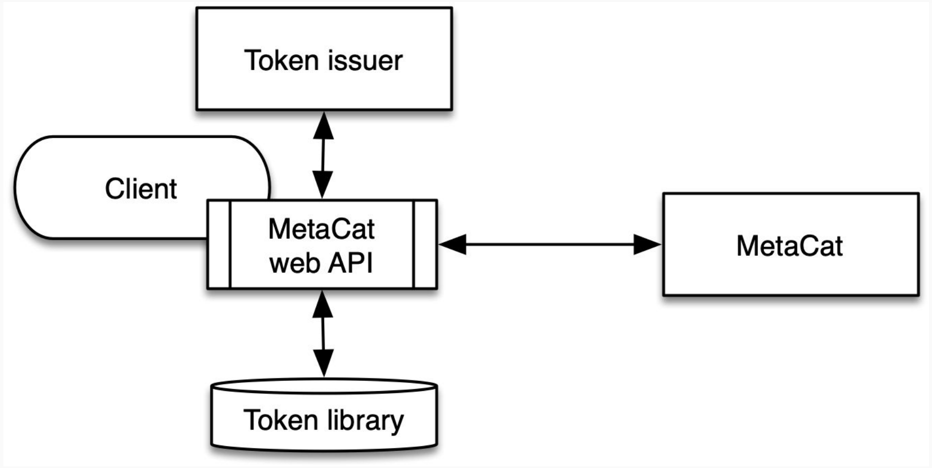
## Tokens and Client Authentication



To accept tokens from other issuers:

- Use public key encryption
- Have access to public keys of trusted issuers
- Trust authentication, authorization info stored in the token
    - Perhaps selectively based on the issuer

# Client side components



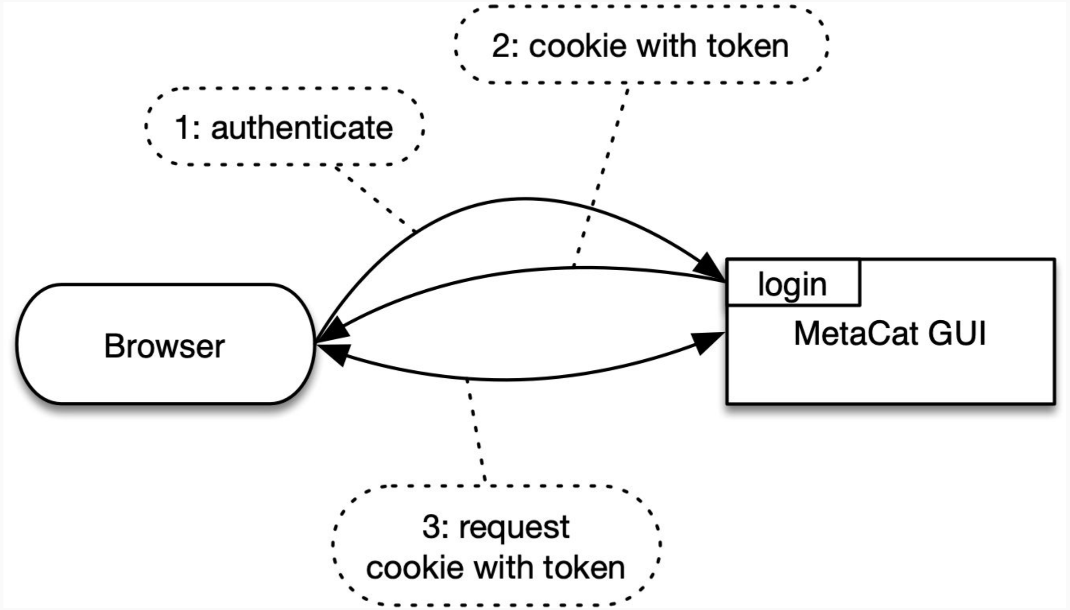
Token Library structure:

MetaCat server URL -> token
MetaCat server URL -> token
...

Functions:

- Add token (URL, token)
- Get token (URL) -> token
- Export token -> text
- Import token <- text
- (purge expired - hidden from user)

# GUI client



On successful authentication, MetaCat GUI gives the client a cookie with the token

Cookie expiration = token expiration

Authentication (for now):

- password, local or LDAP
- X.509
- SSO is not implemented yet

HTTPS is used to secure the token transfer over the wire

## Summary

- Token is the result of successful authentication using one of many implemented authentication mechanisms
    - Once the client is authenticated, their MetaCat access rights are determined by set of user's roles and object ownership
- JSON Web Token (JWT) as defined by IETF in RFC7519
    - PyJWT - Python implementation
    - Symmetric or public key encryption/signature
- Token claims used so far (all standard JWT):
    - Issuer
    - Subject
    - Expiration
    - Issued
    - Not before
    - Token ID

## Summary (2)

- Token includes:
  - Client identity (username as the value of the “subject” claim)
  - May include authorization information (currently does not)
    - User roles in addition to or instead of those written in the MetaCat DB
- Assumptions:
  - Public keys of trusted Token Issuers are available
  - MetaCat client API has access to the tokens, can find token for specific MetaCat server