

# Beyond X.509: Token-based Authentication and Authorization for HEP

**Andrea Ceccanti**, Marco Caberletti, Enrico Vianello, Francesco Giacomini  
INFN CNAF

CHEP 2018

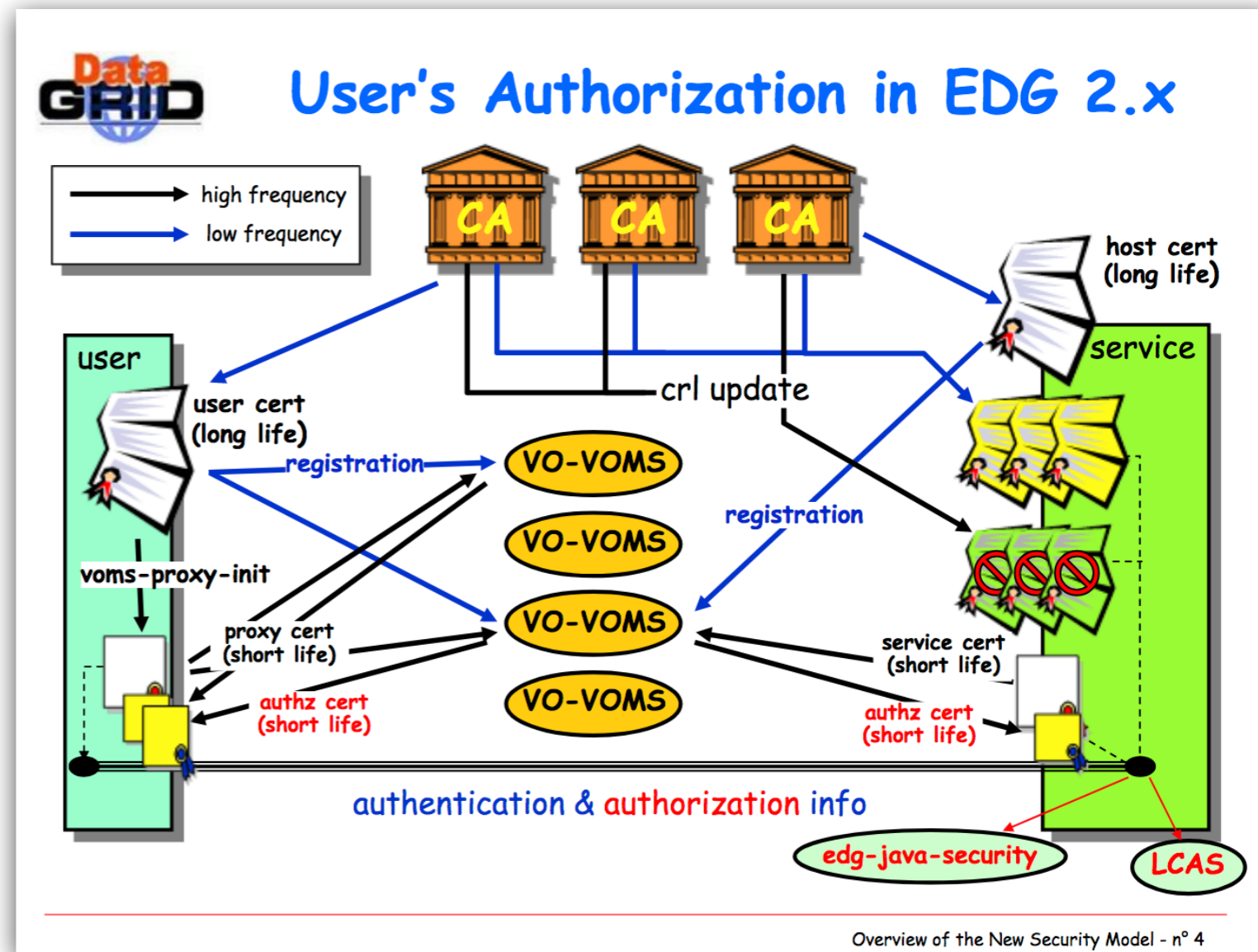
Sofia, July 12th 2018



# The current WLCG AAI

In operation since ~2003,  
and still working nicely:

- **X.509 trust fabric** provided by **IGTF** (tells services which CAs are trusted)
- **X.509 certificates** provided to users for authentication
- **Proxy certificates** for Single Sign-On (SSO) and delegation
- **VOMS attribute certificates** for attribute-based authorization (issued and signed by VO-scoped VOMS servers)



Slide by Ákos Frohner

# Current WLCG AAI: the weak points

## Usability

- X.509 certificates are difficult to handle for users
- VOMS does not work in browsers

## Inflexible authentication

- Only one authentication mechanism supported: X.509 certificates
- Hard to integrate identity federations

## Authorization tightly bound to authentication mechanism

- VOMS attributes are inherently linked to an X.509 certificate subject

## Ad-hoc solution

- We had to invent our own standard and develop ad-hoc libraries and central services to implement our own AAI

**Can we do better today?**

# A novel AAI for WLCG: main challenges

## Authentication

- **Flexible**, able to accommodate various authentication mechanisms
  - X.509, username & password, EduGAIN, social logins (Google, GitHub), ORCID, ...

## Identity harmonization & account linking

- Harmonize multiple identities & credentials in a single account, providing a **persistent identifier**

## Authorization

- **Orthogonal** to authentication, **attribute** or **capability-based**

## Delegation

- Provide the ability for **services to act on behalf of users**
- Support for **long-running applications**

## Provisioning

- Support provisioning/de-provisioning of identities to services/relying resources

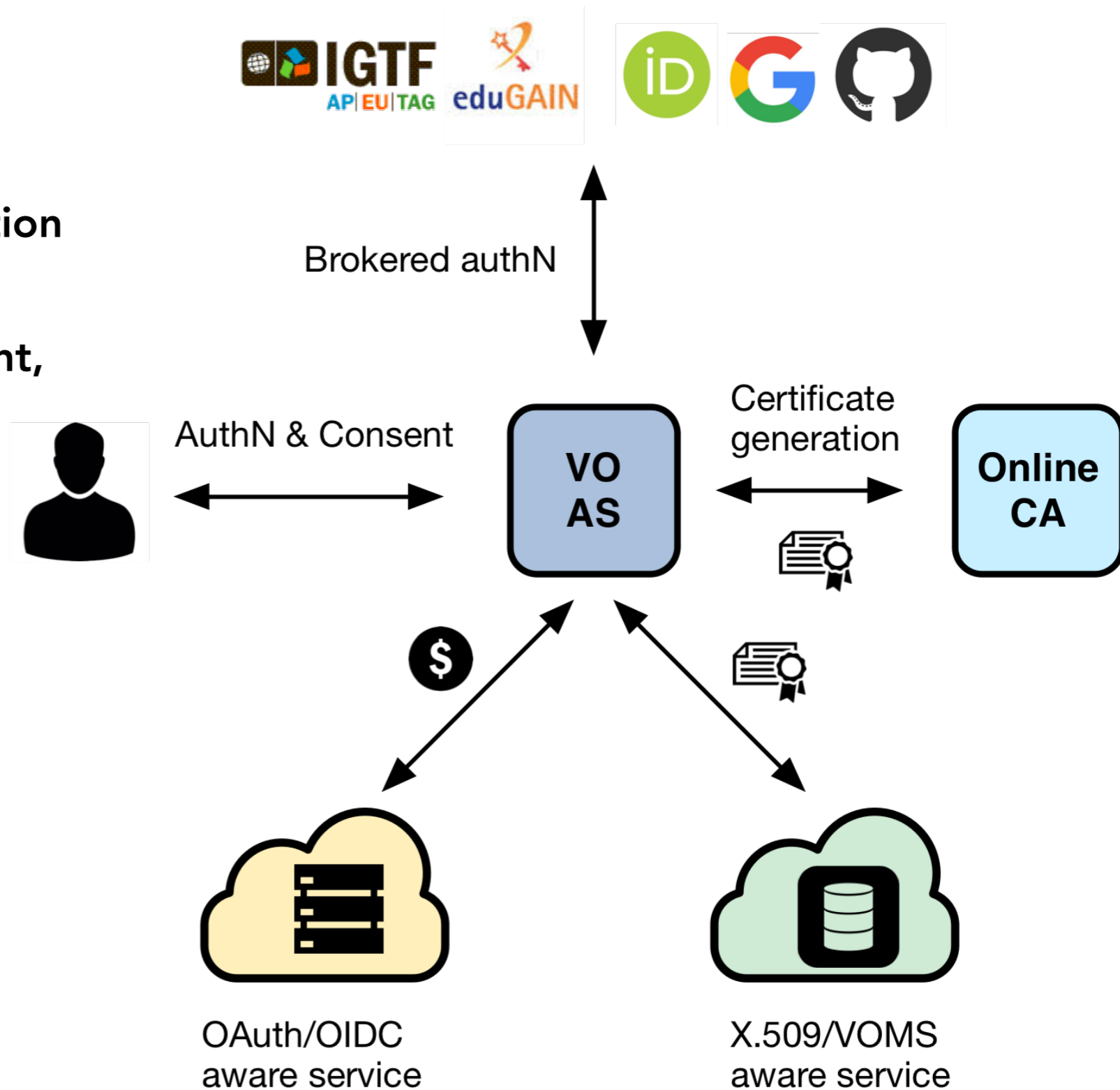
## Token translation

- Enable **integration with legacy services through controlled credential translation**

# A token-based AAI for WLCG

Introduce a central VO-scoped service that

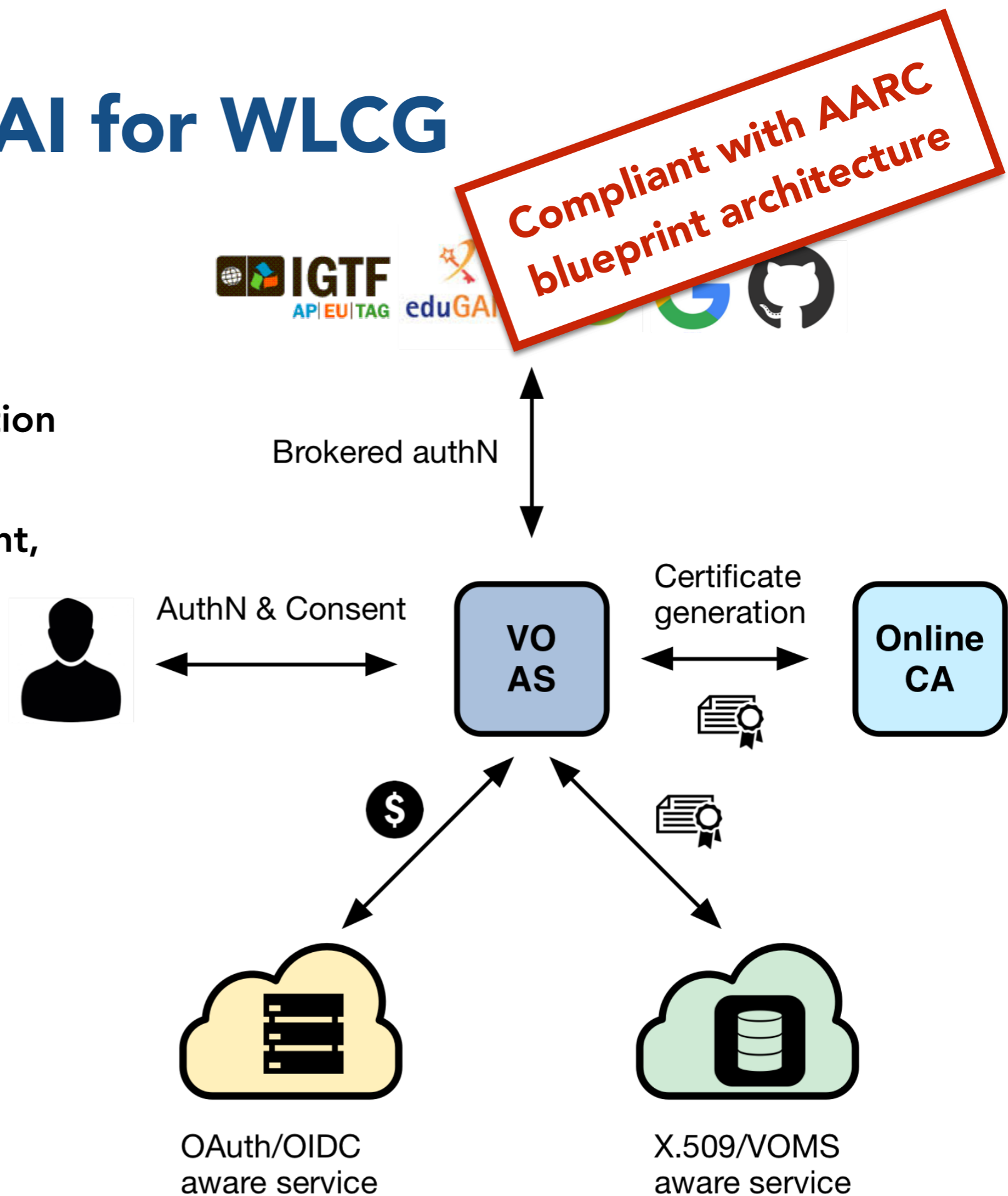
- supports **multiple authentication mechanisms**
- provides users with a **persistent, VO-scoped** identifier
- exposes **identity information, attributes** and **capabilities** to services via **JWT** tokens and standard **OAuth & OpenID Connect** protocols
- can integrate existing **VOMS**-aware services
- supports **Web** and **non-Web access, delegation** and **token renewal**



# A token-based AAI for WLCG

Introduce a central VO-scoped service that

- supports **multiple authentication mechanisms**
- provides users with a **persistent, VO-scoped** identifier
- exposes **identity information, attributes** and **capabilities** to services via **JWT** tokens and standard **OAuth & OpenID Connect** protocols
- can integrate existing **VOMS**-aware services
- supports **Web** and **non-Web access, delegation** and **token renewal**



# **Enabling technologies: an overview**



# Enabling technologies in one slide

## OAuth 2.0

- a standard framework for **delegated authorization**
- widely adopted in industry



## OpenID Connect

- an **identity layer** built on top of OAuth 2
- "OAuth-based authentication done right"



## JSON Web Tokens (JWTs)

- a **compact, URL-safe** means of representing **claims** to be transferred between two (or more) parties

```
{
  "sub": "e1eb758b-b73c-4761-bfff-adc793da409c",
  "aud": "iam-client test",
  "iss": "https://iam-test.indigo-datacloud.eu/",
  "exp": 1507726410,
  "iat": 1507722810,
  "jti": "39636fc0-c392-49f9-9781-07c5eda522e3"
}
```



# OAuth: an example delegated authorization flow

Authorization Server

Resource Server



Resource  
Owner

**Example:**

Link a user Twitter account to his Facebook account, so that when he tweets something, the tweets are visible for his Facebook friends



Client

# OAuth: an example delegated authorization flow

Authorization Server

Resource Server



Resource  
Owner

**Resource owner:**

The user who owns his  
Facebook page and  
drives the delegated  
authorization process



Client

# OAuth: an example delegated authorization flow

Authorization Server  
Resource Server



Resource  
Owner

**Client:**

Twitter, which will be granted permission to post on Facebook on behalf of the user when the user tweets something



**Client**

# OAuth: an example delegated authorization flow



**Client**

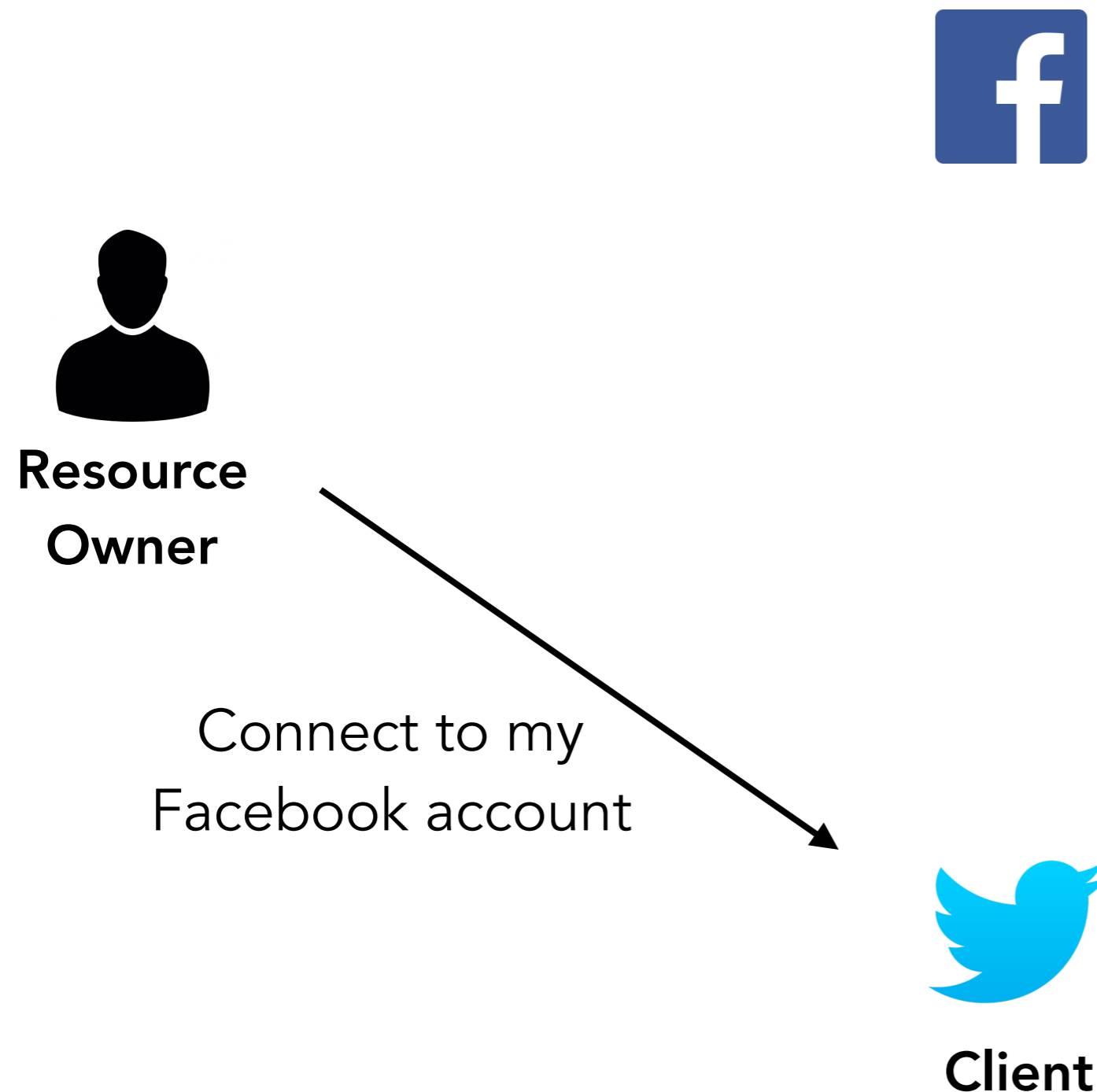
## **Authorization Server:**

Facebook AS will authenticate the user and issue tokens to the client (Twitter) after having obtained a consent from the Resource Owner

## **Resource Server:**

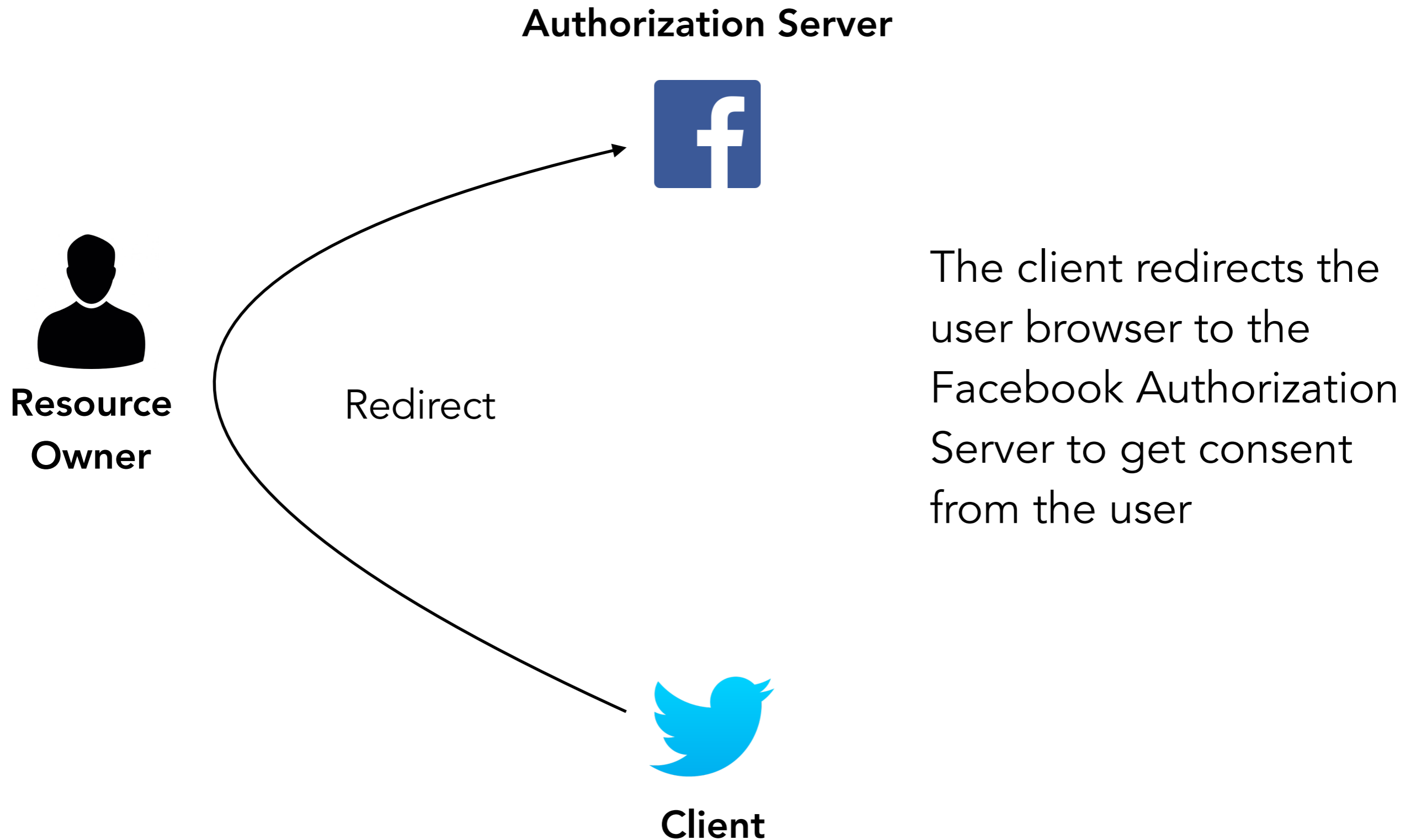
Facebook RS (APIs) will grant access only to those clients presenting a valid token, and limit actions according to the scopes linked to the token

# OAuth: an example delegated authorization flow



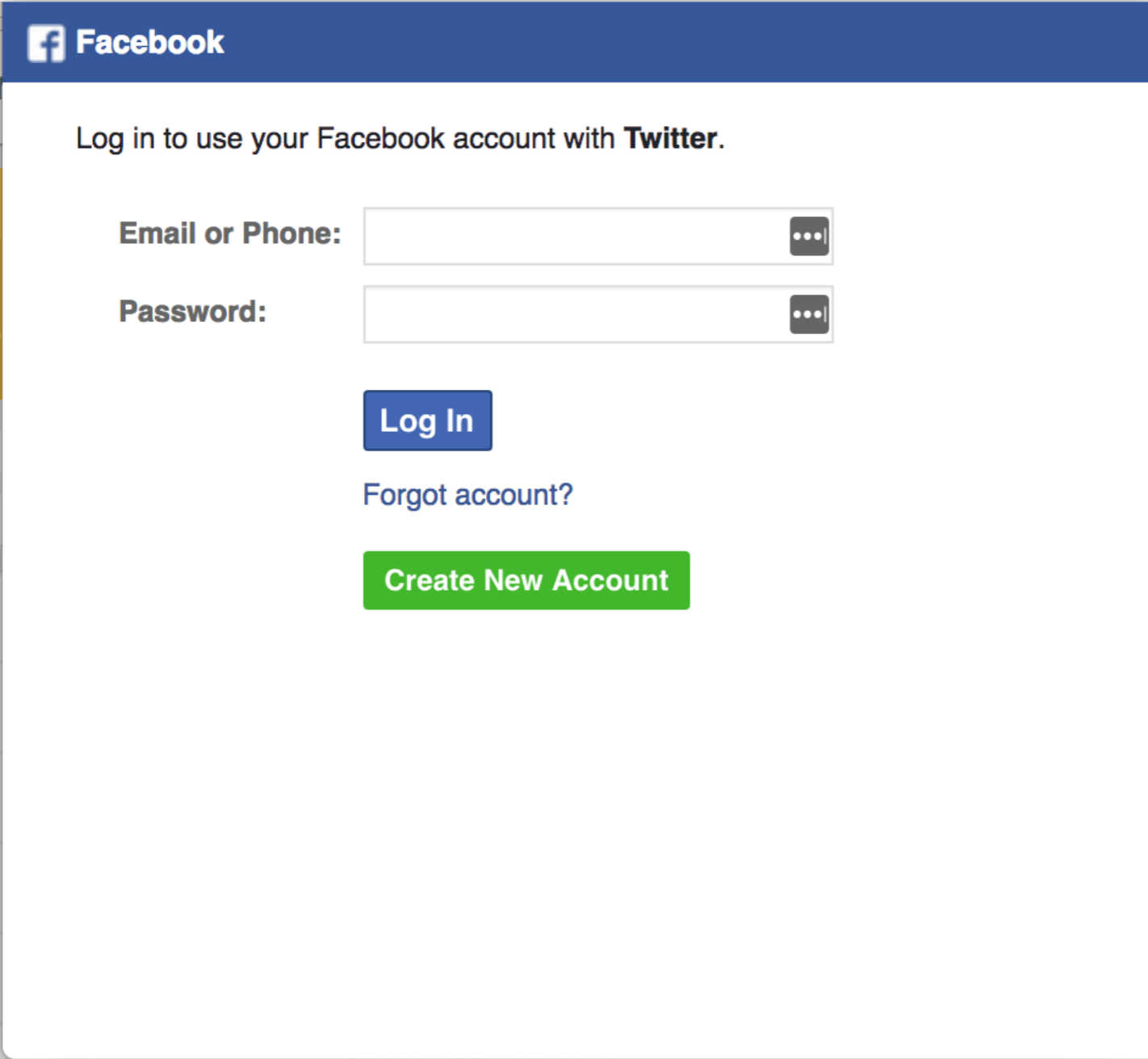
The user (aka Resource Owner) requests that his Twitter account is connected to his Facebook page

# OAuth: an example delegated authorization flow



# OAuth: an example delegated authorization flow

## Authorization Server



The image shows a screenshot of a Facebook login page. At the top, there is a blue header with the Facebook logo and the word "Facebook". Below the header, the text reads "Log in to use your Facebook account with **Twitter**." There are two input fields: "Email or Phone:" and "Password:". Below the "Email or Phone:" field is a blue "Log In" button. Below the "Log In" button is a link that says "Forgot account?". Below the "Forgot account?" link is a green "Create New Account" button. To the left of the login form, there is a black silhouette of a person's head and shoulders. Below the silhouette, the text "Reso" and "Owi" is visible, likely part of a larger text or diagram.

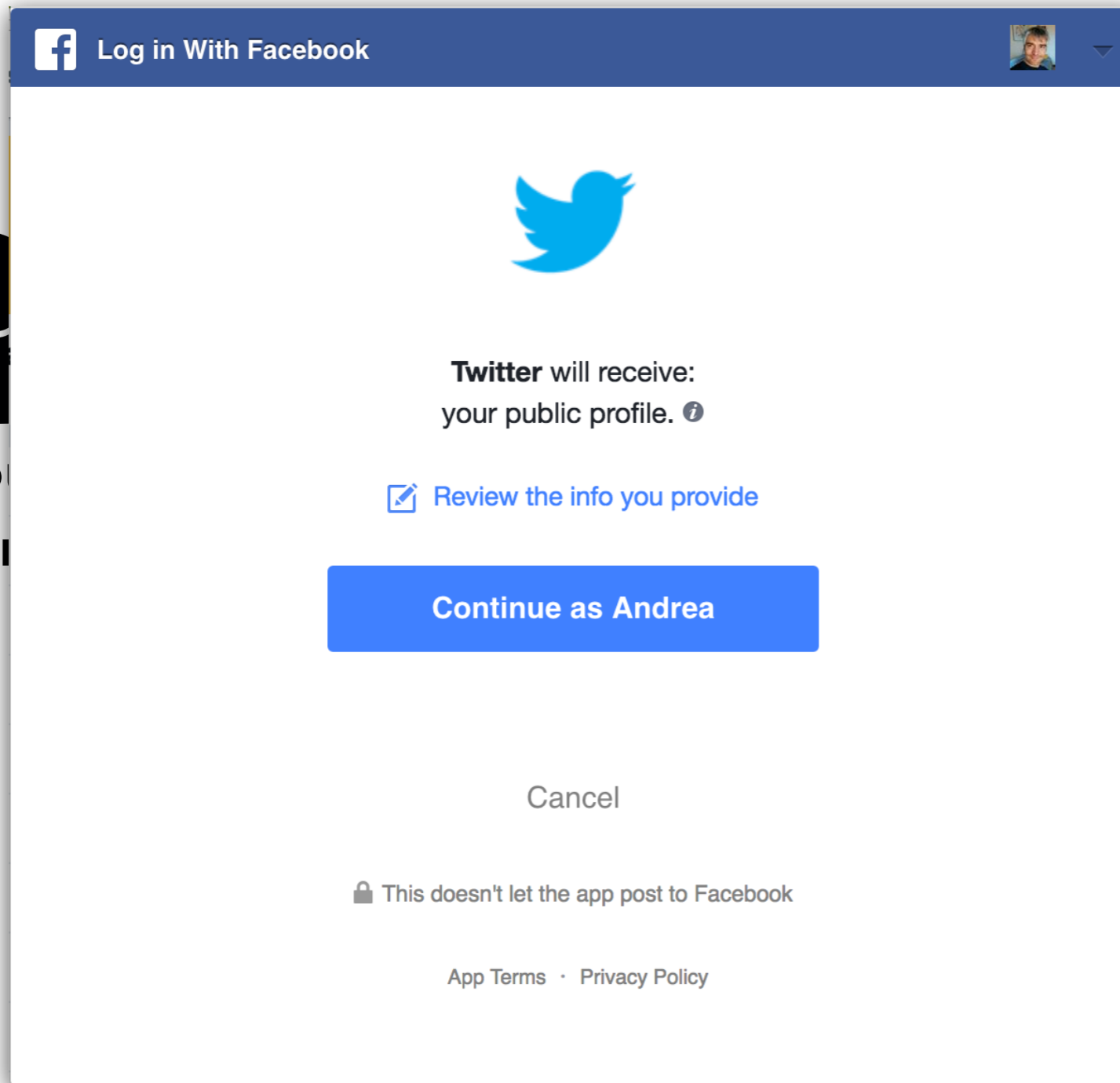
The user does not have an active login session @ Facebook, so the Facebook authorization server asks for authentication



# OAuth: an example delegated authorization flow

## Authorization Server

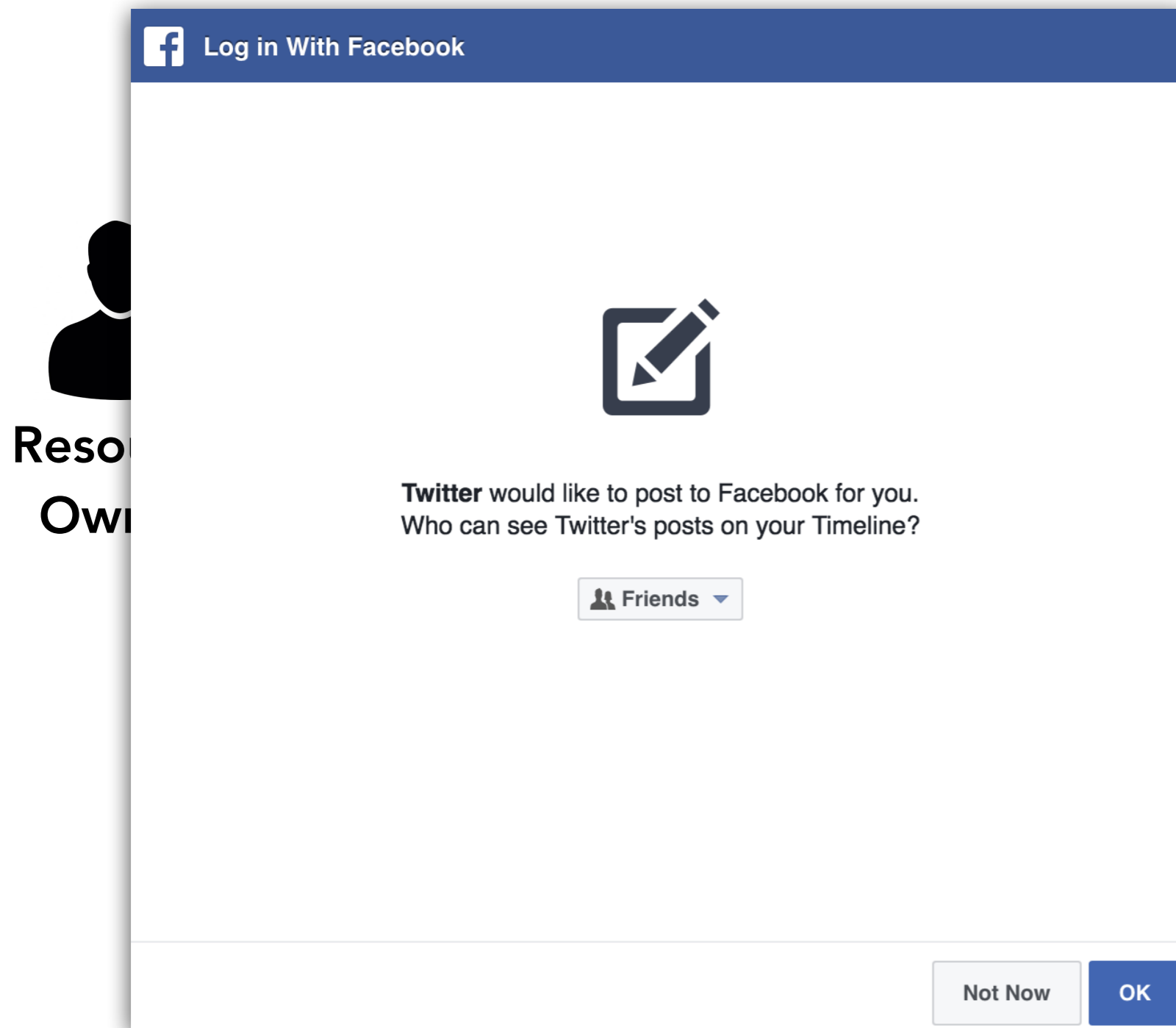
Resource Owner



Once the user is authenticated, Facebook informs the user that some of his profile information will be shared with Twitter. In order to proceed the user has to approve, i.e. to give his **consent**.

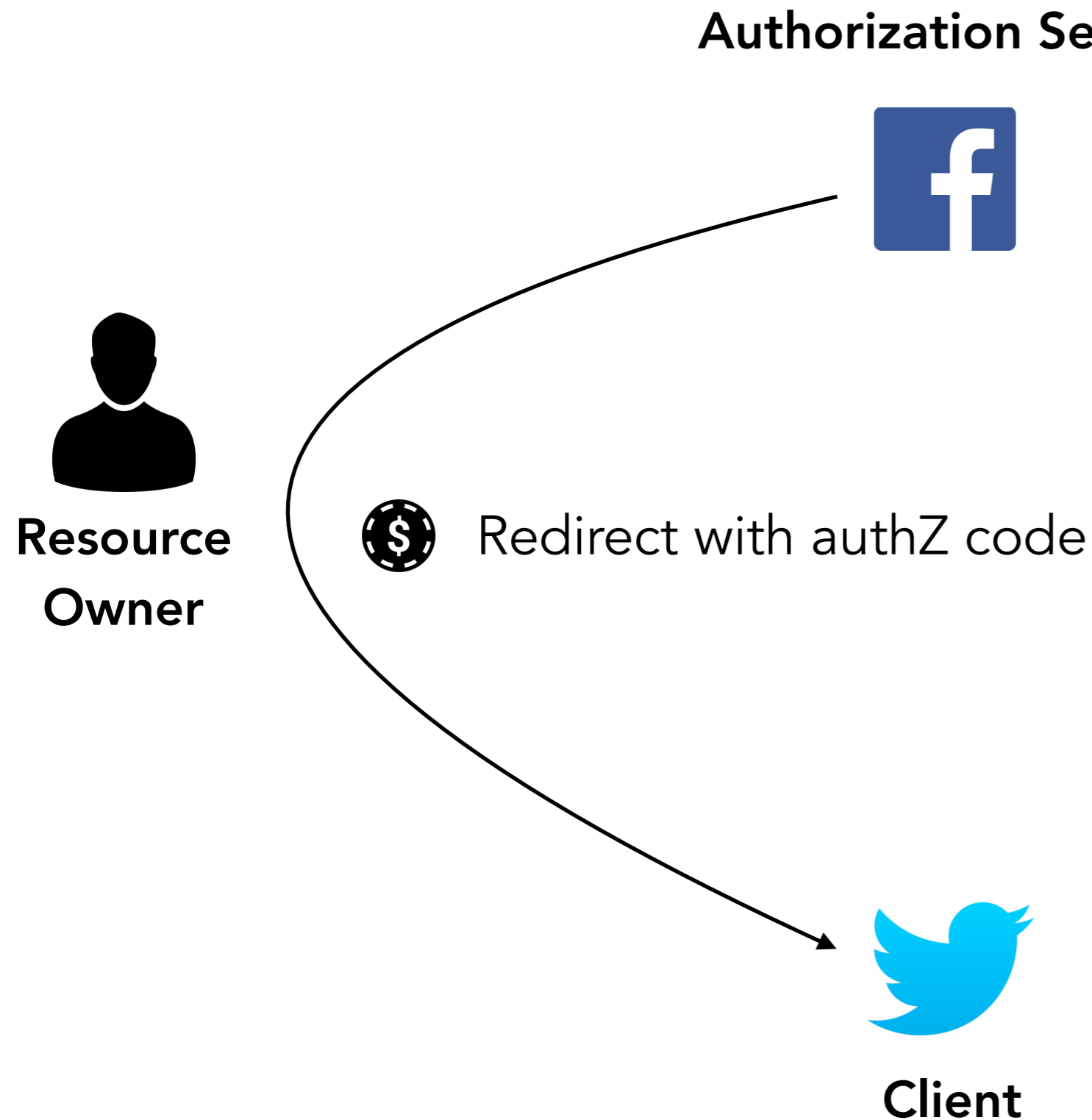
# OAuth: an example delegated authorization flow

## Authorization Server



Twitter also requested the ability to write on the user Facebook page, so Facebook asks for consent also for this.

# OAuth: an example delegated authorization flow



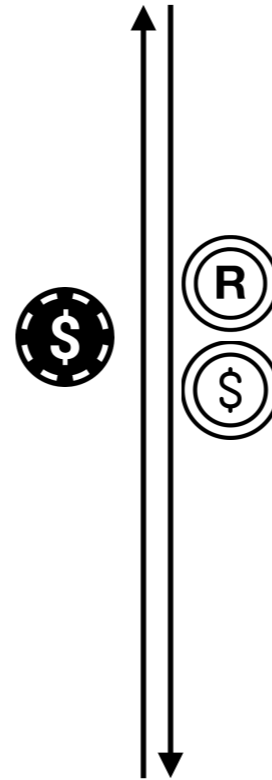
When consent is obtained from the resource owner, Facebook redirects the user to Twitter sending an **authorization code** as a parameter of the HTTP redirect

# OAuth: an example delegated authorization flow

Authorization Server



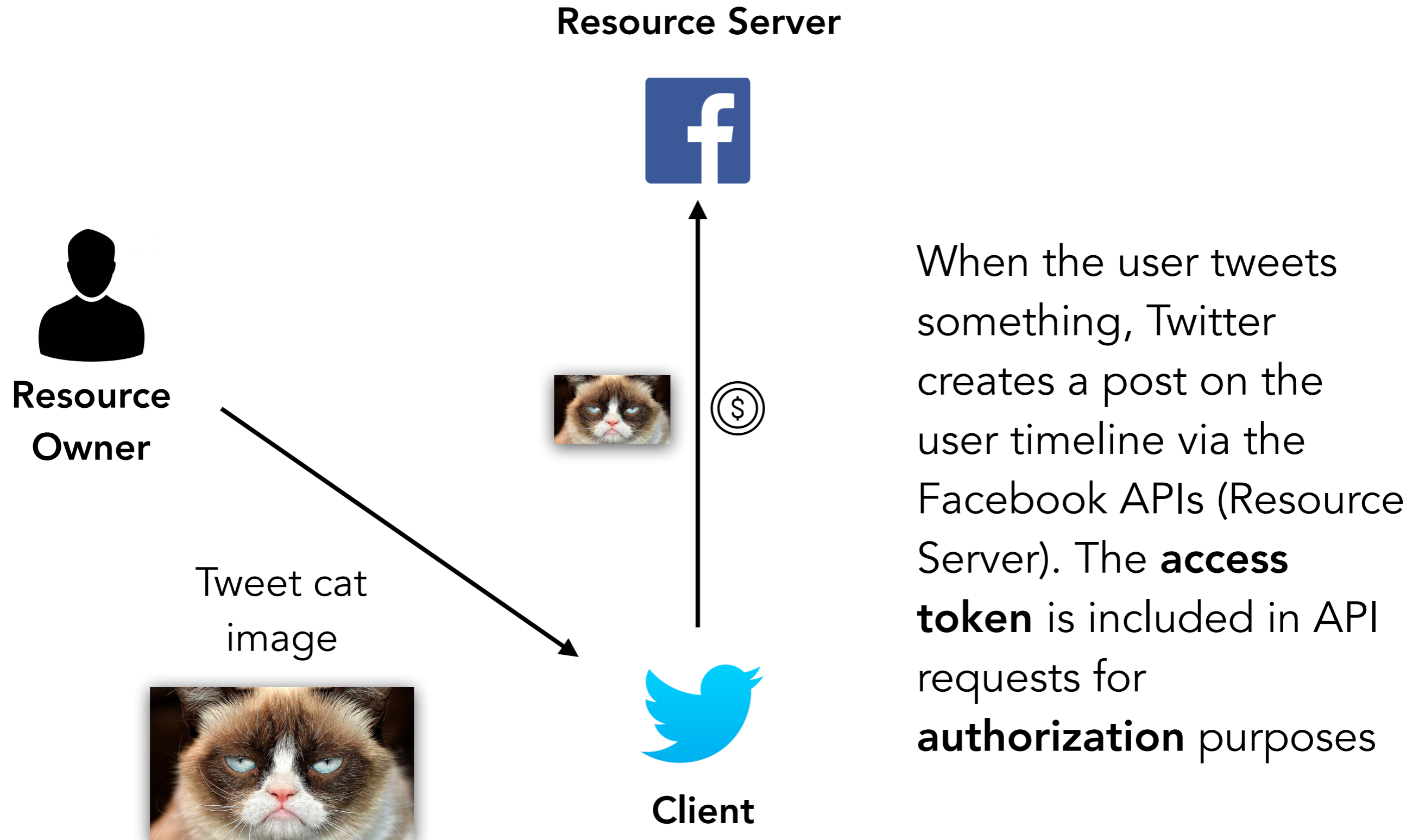
Resource  
Owner



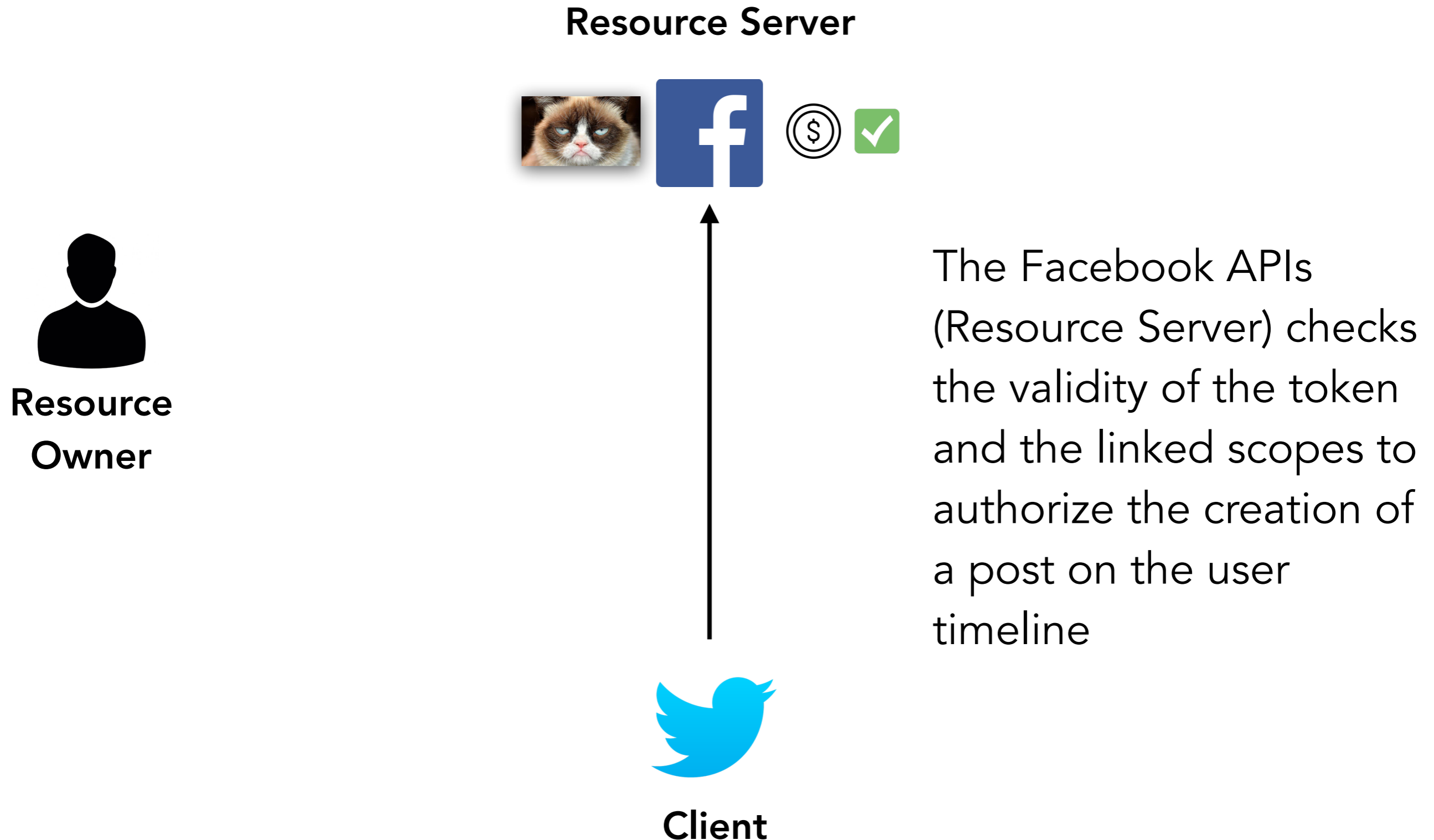
Client

Twitter then exchanges the **authorization code** with a short-lived **access token** (and a long-lived **refresh token**). This is a direct message exchange between Twitter and the Facebook AS

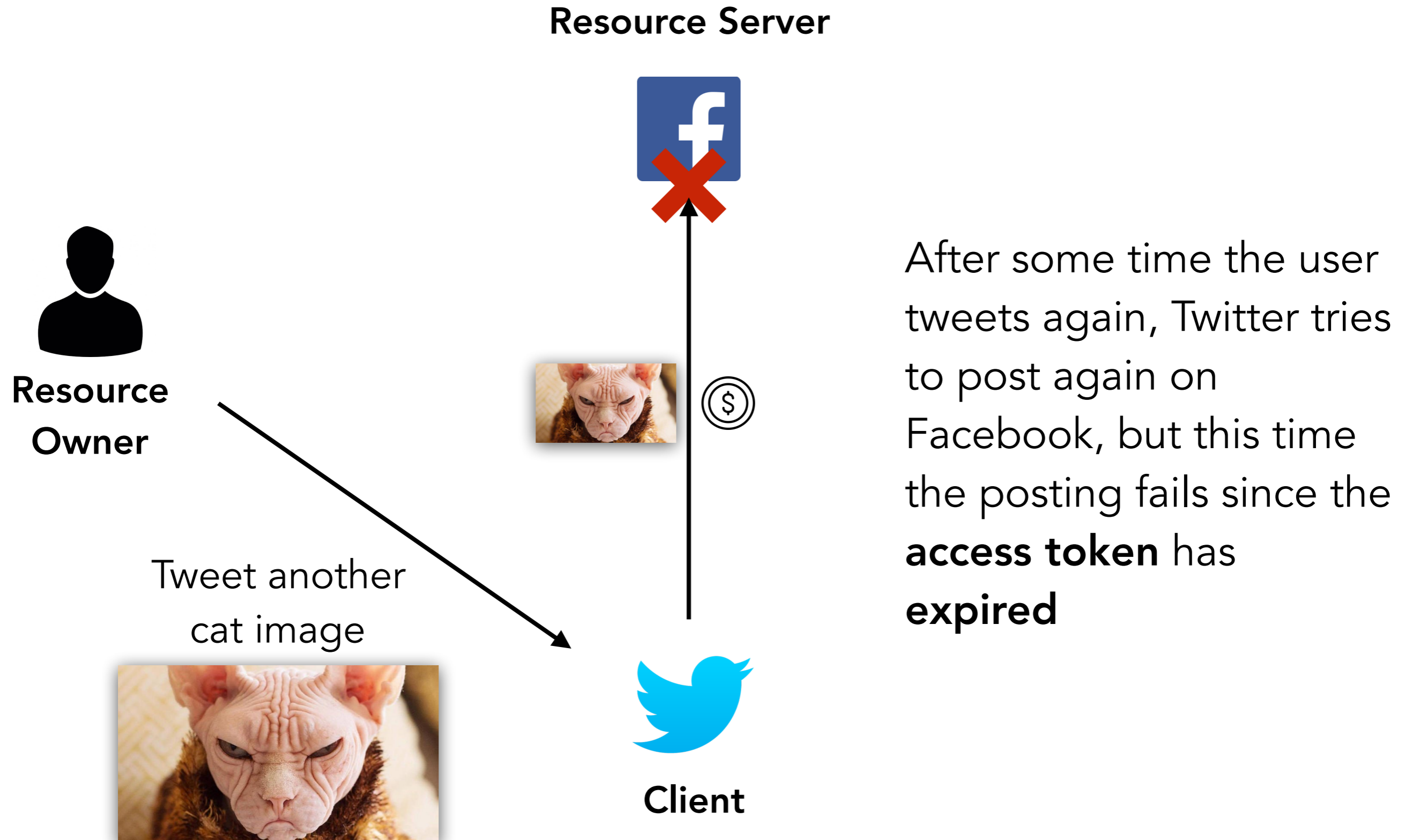
# OAuth: an example delegated authorization flow



# OAuth: an example delegated authorization flow



# OAuth: an example delegated authorization flow





# OAuth: an example delegated authorization flow

Authorization Server

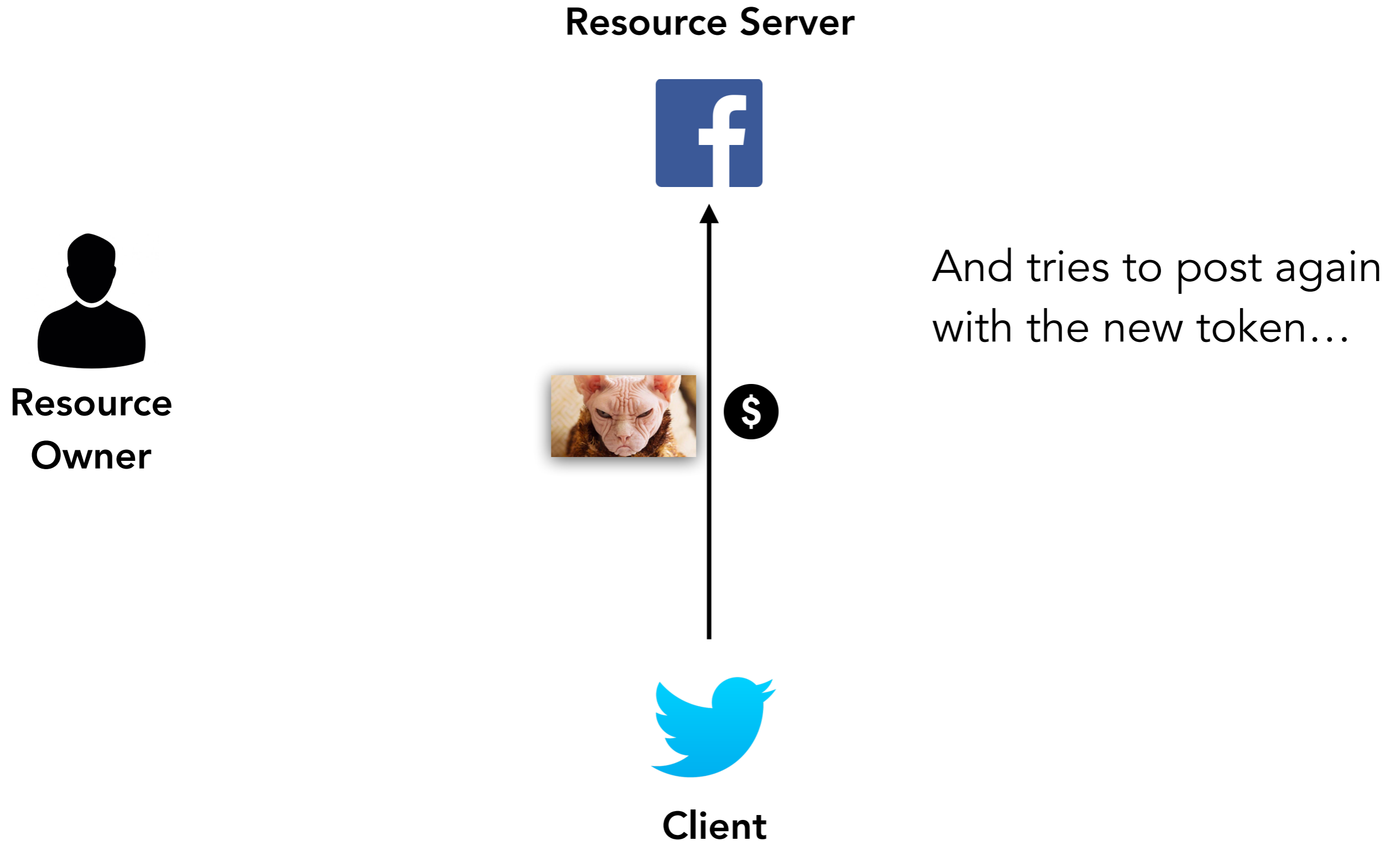


Client

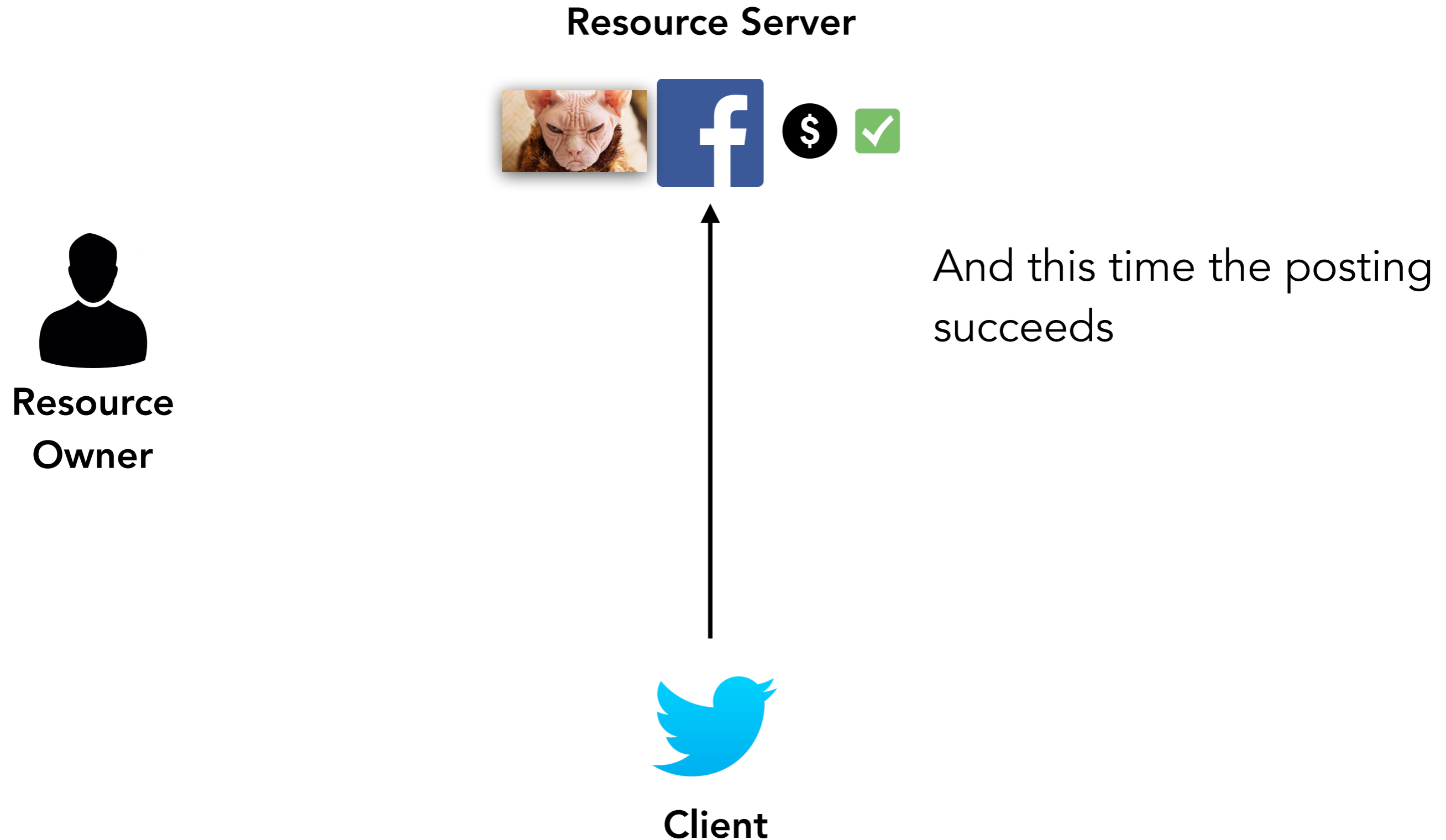
Twitter then presents a **refresh token** to the Facebook AS to obtain a new access token



# OAuth: an example delegated authorization flow



# OAuth: an example delegated authorization flow



# OpenID Connect: an identity layer for OAuth

OAuth is a **delegated authorization** protocol

- an **access token** states the **authorization rights** of the client application presenting the token to access some resources



OpenID Connect extends OAuth to provide a standard **identity layer**

- i.e. information about **who the user is** and **how it was authenticated** via an additional **ID token (JWT)** and a dedicated **user information query endpoint** at the OpenID Connect Identity provider
- provides ability to establish **login sessions** (SSO)



# JSON Web Tokens (JWT)

**JSON Web Token** (JWT) is an open standard that defines a compact, self-contained way of securely transmitting information between parties as a JSON object

JWTs are typically **signed** and, if confidentiality is a requirement, can be **encrypted**.

## Header

```
{  
  "kid": "rsa1",  
  "alg": "RS256"  
}
```

## Body

```
{  
  "sub": "e1eb758b-b73c-4761-bfff-adc793da409c",  
  "iss": "https://iam-test.indigo-datacloud.eu/",  
  "exp": 1482163788,  
  "iat": 1482160188,  
  "jti": "e7bcb54c-8f67-4a77-8415-37adeb4b958c"  
}
```

## Signature

```
Qb0fPrha9kp4e7TknXe88  
d8v_9e7V2v2xMAKX10xY4  
M3P1wragAhQmyoVQwq-uk
```

# Why OAuth, OpenID Connect and JWT?

Standard, widely adopted in industry

- Do not reinvent the wheel, reuse existing knowledge and tools, extend when needed

Reduced integration complexity at relying services

- Off-the-shelf libraries and components

Authentication-mechanism agnostic

- The AAI is not bound to a specific authentication mechanism

Distributed verification of access and identity tokens

- It scales

**Back to our token-based AAI...**



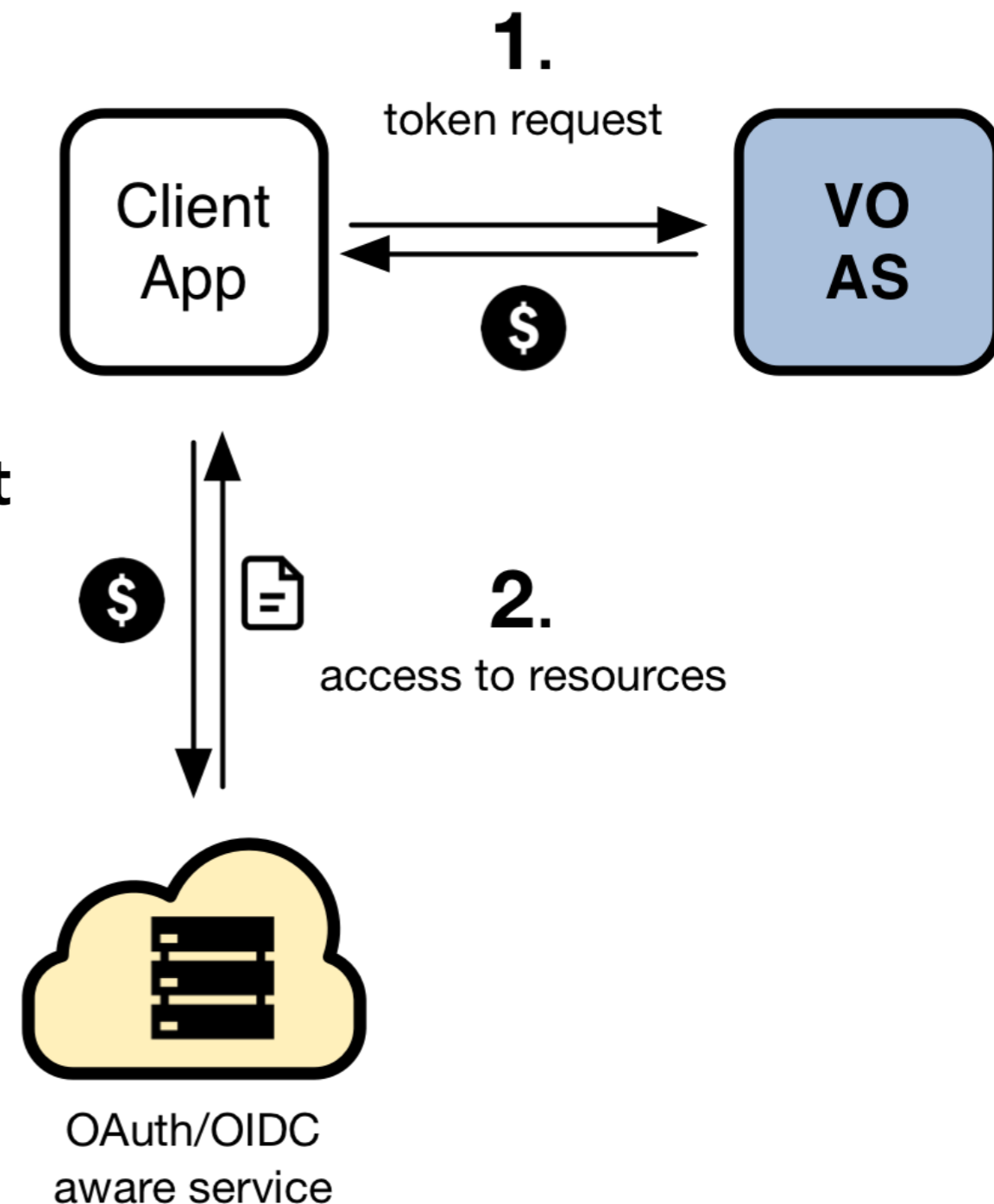
# OAuth and OpenID Connect for WLCG

In order to access resources/services, a **client application** needs an **access token**

The token is obtained from a **VO** (which acts as an OAuth Authorization Server) using standard **OAuth/OpenID Connect** flows

**Authorization** is then **performed at the services** leveraging info extracted from the token:

- **Identity attributes:** e.g., groups
- **OAuth scopes:** authZ labels that are linked to access tokens at token creation time



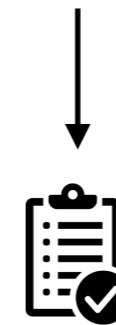
# Attribute-based vs Scope-based Authorization

**Attribute-based authorization:** the token brings information about attribute ownership (e.g., groups/role membership), the service maps these attributes to a local authorization policy

**Scope-based authorization:** the token brings information about which actions should be authorized at a service, the service needs to understand these capabilities and honor them. The authorization policy is managed at the VO level

token claims

```
{  
  "iss": "https://cms.wlwg.example",  
  ...  
  "groups": "/cms"  
}
```



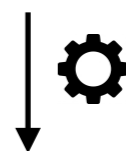
policy



authZ  
decision

token claims

```
{  
  "iss": "https://cms.wlwg.example",  
  ...  
  "scope": "read:/ upload:/store"  
}
```



authZ  
decision

# INDIGO Identity and Access Management service

## Flexible authentication support

- (SAML, X.509, OpenID Connect, username/password, ...)

## Account linking

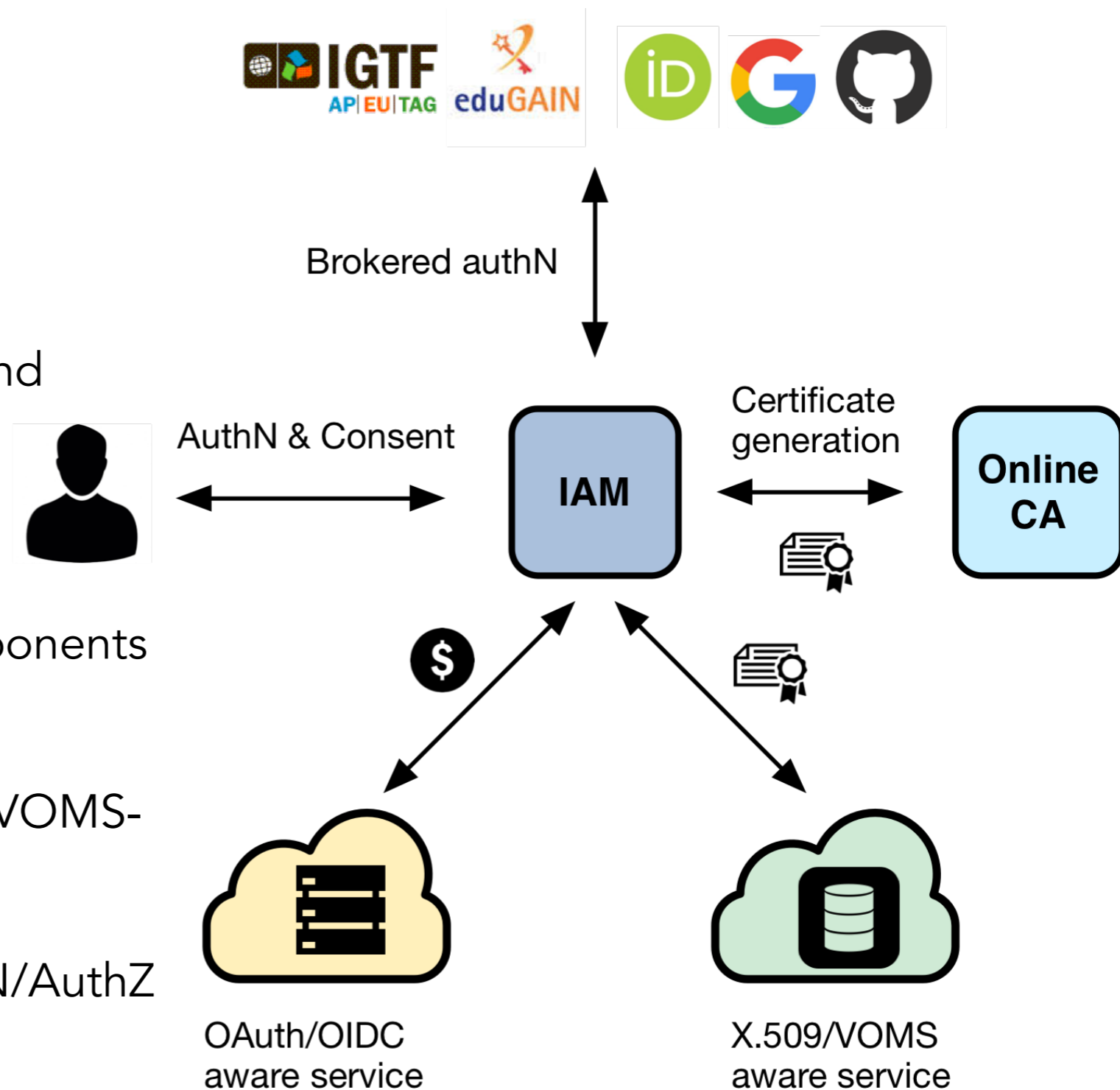
**Registration service** for moderated and automatic user enrollment

**Enforcement of AUP acceptance**

**Easy integration** in off-the-shelf components thanks to **OpenID Connect/OAuth**

**VOMS support**, to integrate existing VOMS-aware services

**Self-contained**, comprehensive AuthN/AuthZ solution



# INDIGO Identity and Access Management service

## Flexible authentication support

- (SAML, X.509, OpenID Connect, username/password, ...)

## Account linking

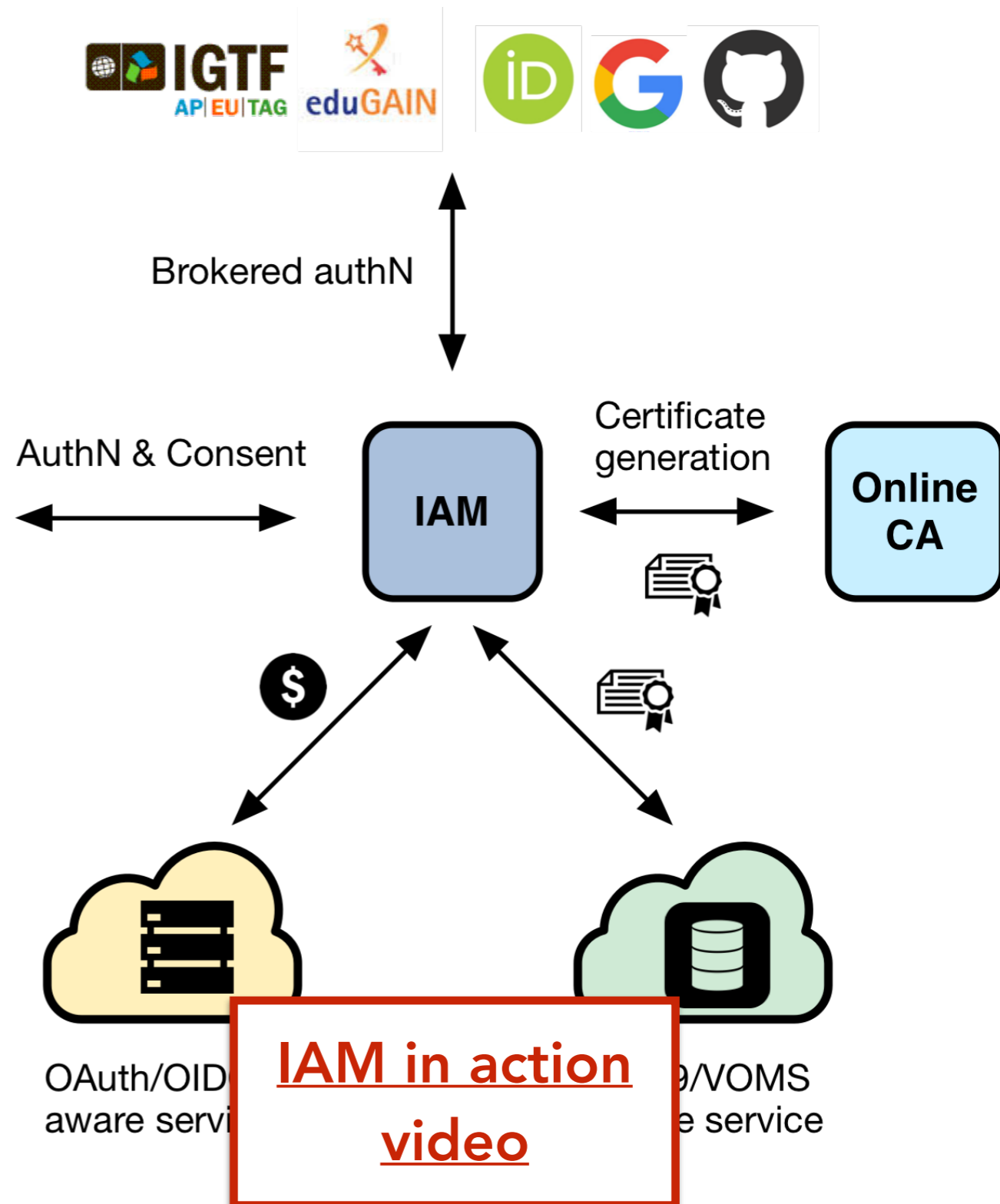
**Registration service** for moderated and automatic user enrollment

## Enforcement of AUP acceptance

**Easy integration** in off-the-shelf components thanks to **OpenID Connect/OAuth**

**VOMS support**, to integrate existing VOMS-aware services

**Self-contained**, comprehensive AuthN/AuthZ solution



# IAM deployment model

An IAM instance is deployed for a **community** of users sharing resources, the good old **Virtual Organization (VO)** concept

Client applications and services are integrated with this instance via **standard OAuth/OpenID Connect**

The IAM Web appearance can be **customized** to include a **community logo**, **AUP** and **privacy policy** document

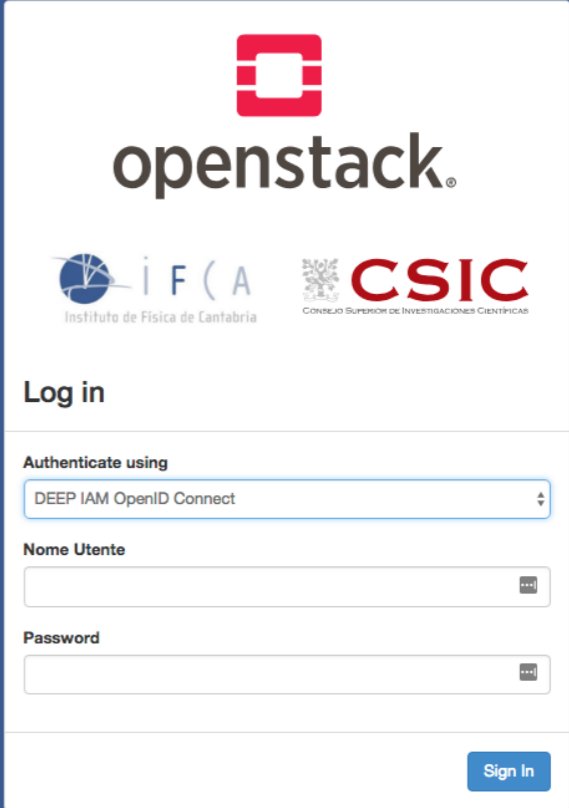
The image displays three screenshots of IAM login pages, stacked vertically. The top screenshot shows the INFN CHNet login page with the logo and text 'Istituto Nazionale di Fisica Nucleare Cultural Heritage Network' and 'Welcome to chnet'. It features a 'Username' field and a 'Password' field. The middle screenshot shows the deep Hybrid DataCloud login page with the logo and text 'Welcome to deep-hdc'. The bottom screenshot shows the WLCG Worldwide LHC Computing Grid login page with the logo and text 'Welcome to wlcg-authz-wg'. It features a 'Username' field, a 'Password' field, a 'Sign in' button, a 'Forgot your password?' link, a 'Sign in with Google' button, and a 'Register a new account' button. Below the login fields, it displays the authenticated user information: 'You have been successfully authenticated as CN=Andrea Ceccanti aceccant@infn.it,O=Istituto Nazionale di Fisica Nucleare,C=IT,DC=tcs,DC=terena,DC=org' and a note: 'This certificate is not linked to any account in this organization'.

# Easy integration with services

Standard OAuth/OpenID Connect enable **easy integration** with off-the-shelf services and libraries.

We have successfully integrated IAM with minimal effort with:

- Openstack
- Atlassian JIRA & Confluence
- Kubernetes
- Moodle
- Rocketchat
- Grafana
- JupyterHub



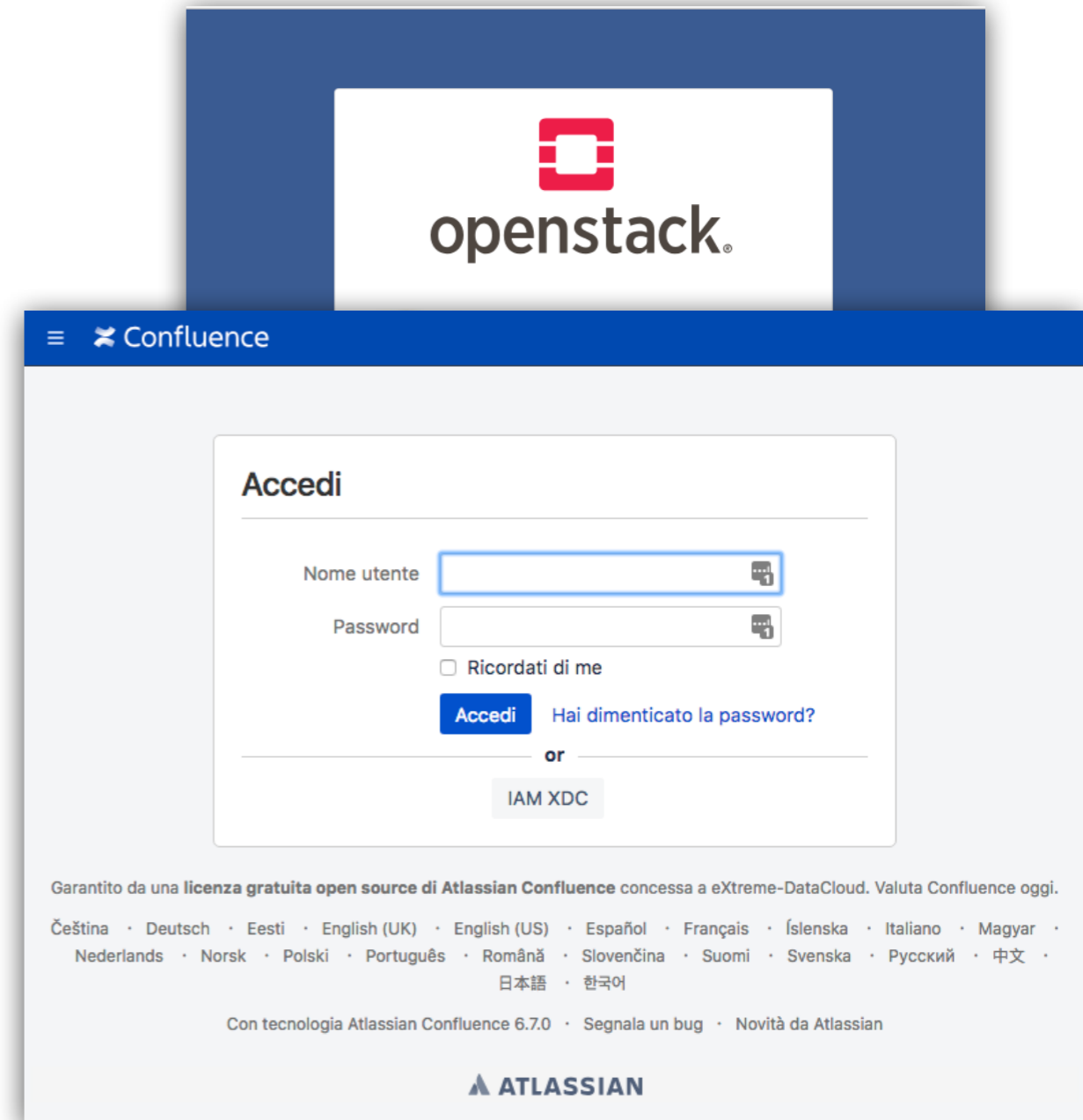
The screenshot shows a login interface for OpenStack. At the top, there is the OpenStack logo and the text 'openstack.'. Below this, there are logos for 'IFCA Instituto de Física de Cantabria' and 'CSIC CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS'. The main section is titled 'Log in'. It contains a dropdown menu labeled 'Authenticate using' with the selected option 'DEEP IAM OpenID Connect'. Below the dropdown are two input fields: 'Nome Utente' and 'Password', each with a toggle icon. A blue 'Sign In' button is positioned at the bottom right of the form.

# Easy integration with services

Standard OAuth/OpenID Connect enable **easy integration** with off-the-shelf services and libraries.

We have successfully integrated IAM with minimal effort with:

- Openstack
- Atlassian JIRA & Confluence
- Kubernetes
- Moodle
- Rocketchat
- Grafana
- JupyterHub

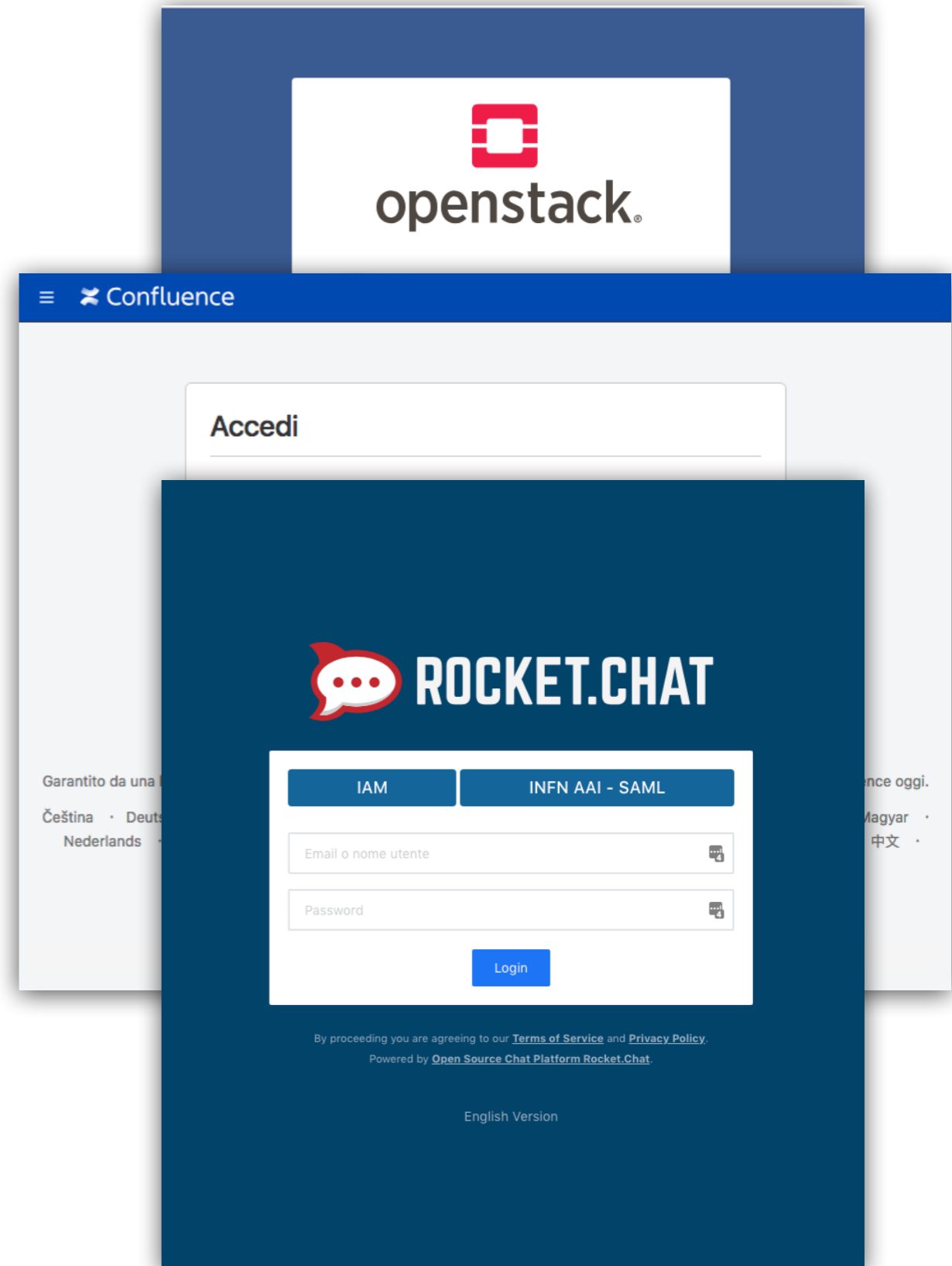


# Easy integration with services

Standard OAuth/OpenID Connect enable **easy integration** with off-the-shelf services and libraries.

We have successfully integrated IAM with minimal effort with:

- Openstack
- Atlassian JIRA & Confluence
- Kubernetes
- Moodle
- Rocketchat
- Grafana
- JupyterHub



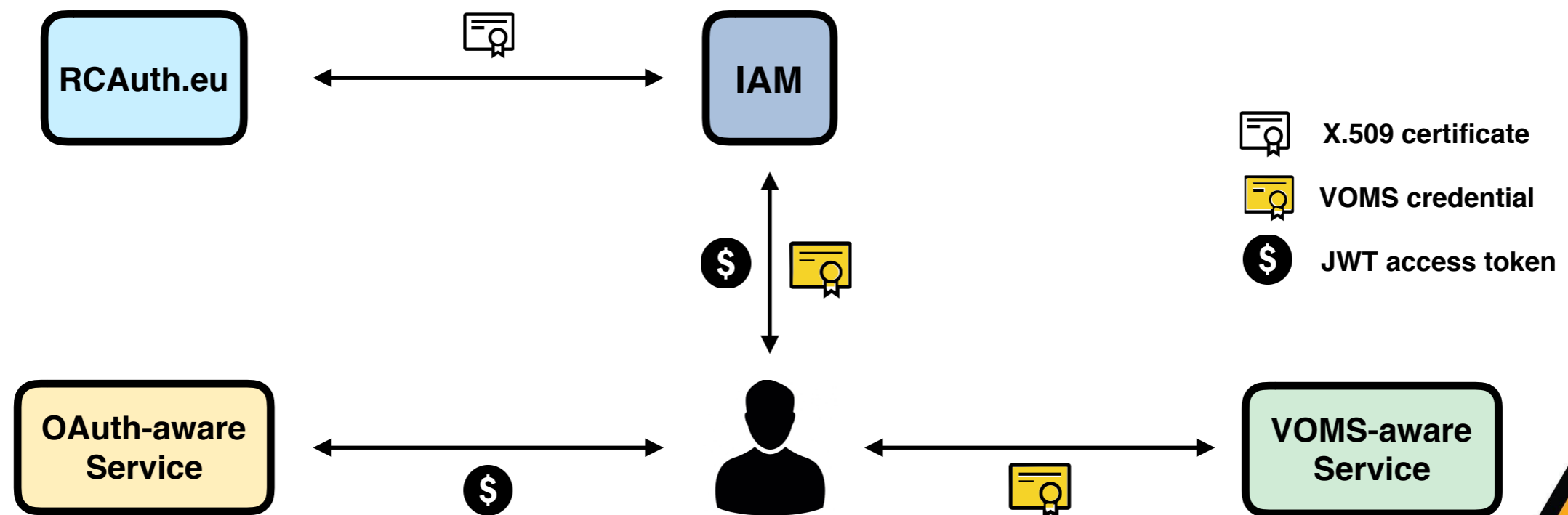


# Seamless transition from an X.509-based AAI

An IAM **VOMS endpoint** exposes authentication and authorization information for an IAM user in the form of a **VOMS attribute certificate, compatible** with existing VOMS clients

Integration with the **RCAuth.eu** online CA allows to generate X.509 certificates on-demand and link them to IAM user memberships

**Consistent authorization** for VOMS and OAuth/OIDC services



# Related initiatives

**EOSC-Hub AAI:** harmonization across Identity solutions (EGI CheckIn, INDIGO IAM, B2Access, ...) for an interoperable EOSC AAI ([CHEP 2018 talk](#))

**SciTokens:** OAuth/JWT profile for capability-based authorization and integration in existing middleware (HTCondor, XRootD, ...) ([CHEP 2018 talk](#))

**dCache:** token-based authorization based on macarons, support for OpenID Connect authentication and initial support for SciTokens ([CHEP 2018 talk](#))

**HTTP LHC Data transfer Ecosystem:** HTTP third-party transfers with token-based authorization ([CHEP 2018 talk](#))

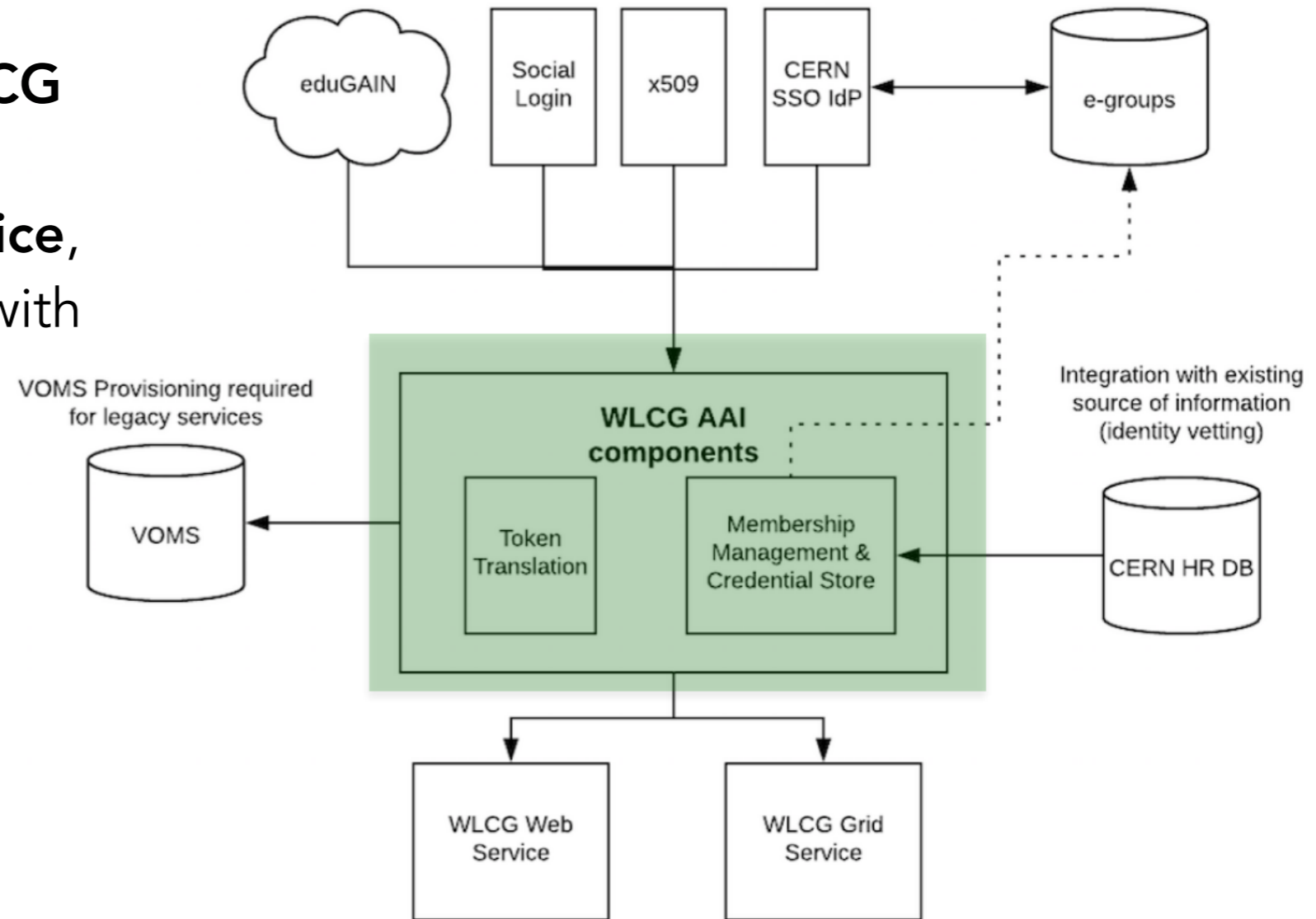
**AARC and FIM4R:** a common AA and policy framework for research communities and recommendations on how to integrate Federated Identity Management ([CHEP 2018 talk](#))

# The WLCG Authorization WG

<https://twiki.cern.ch/twiki/bin/view/LCG/WLCGAuthorizationWG>

Main objectives:

- Design and testing of a **WLCG Membership Management and Token Translation service**, facilitated by pilot projects with the support of AARC
- Definition of a **token-based authentication and authorization profile for WLCG**

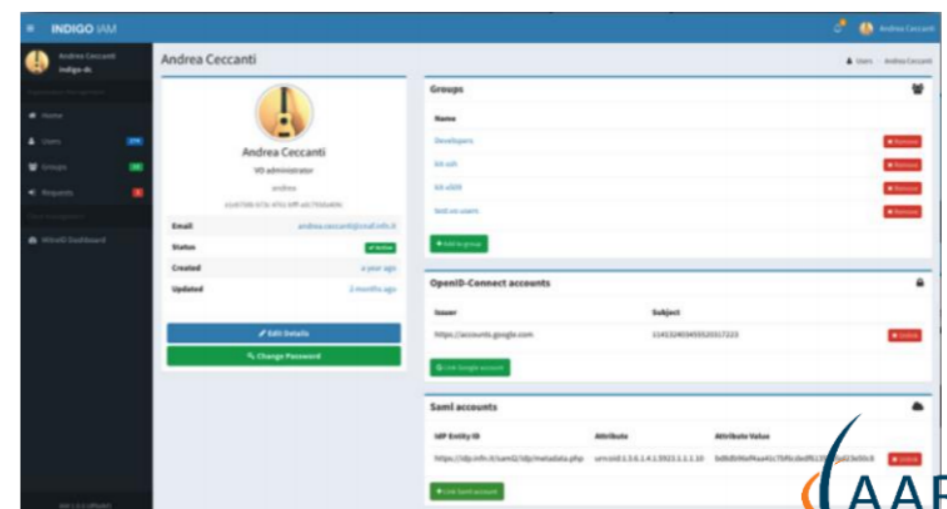
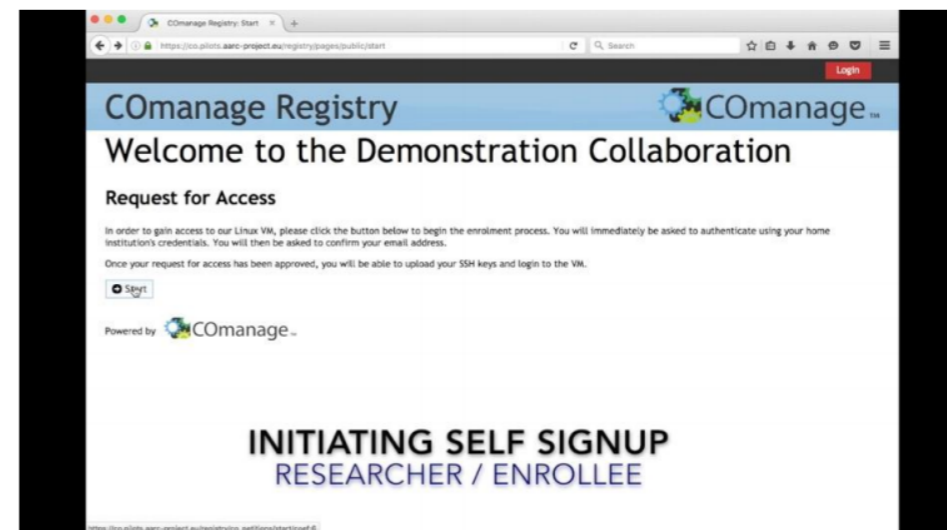


# The WLCG Authorization WG

<https://twiki.cern.ch/twiki/bin/view/LCG/WLCGAuthorizationWG>

## AAI Pilot Projects

- Two solutions appear to meet the majority of requirements
  - EGI Check-in & CManage
  - INDIGO IAM
- Additional integration required for
  - VOMS provisioning & lookup
  - CERN HR DB integration
  - AUP re-signing

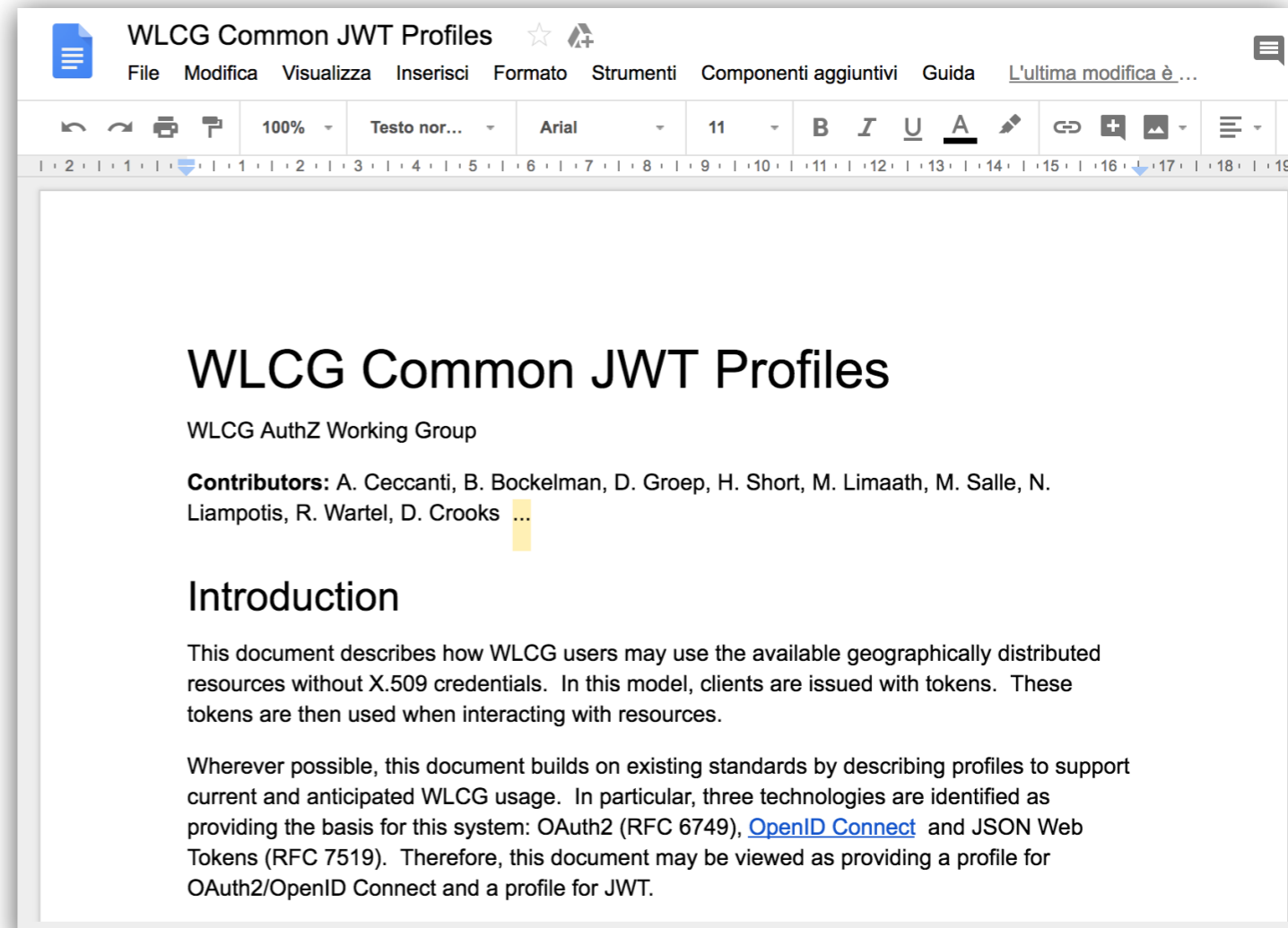


# A common profile for Token-based AuthN/AuthZ

How is **authentication** and **authorization** information encoded in **identity** and **access tokens**?

How is **trust** established between parties exchanging tokens?

What's the recommended **token lifetime**?



Approach:

**rely on existing standards as much as possible,  
extend only when needed**

# Summary

Moving **beyond X.509** certificates and VOMS is recognized as a key challenge for HEP computing to

- improve usability
- simplify the middleware stack thus reducing development and maintenance costs

Convergence across initiatives and research infrastructures on moving towards **standards-based token authentication & authorization**

- based on OAuth, OpenID Connect, JWTs

**INDIGO IAM** represents the **evolution of VOMS** and is one of the solutions under evaluation by the WLCG AuthZ WG that can enable a **smooth transition** between the current and the future token-based WLCG AAI

The **WLCG Authorization WG** is bringing the experts together to define the requirements for this transition, a common profile for token-based authentication and authorization and assess existing solutions

**Thanks for your attention.  
Questions?**

**Backup slides**



# Useful references

IAM @ GitHub: <https://github.com/indigo-iam/iam>

IAM documentation: <https://indigo-iam.github.io/docs>

WLCG Authorization WG: <https://twiki.cern.ch/twiki/bin/view/LCG/WLCGAuthorizationWG>

IAM in action video: <https://www.youtube.com/watch?v=1rZlvJADOnY>

## Contacts:

- [andrea.ceccanti@cnaa.infn.it](mailto:andrea.ceccanti@cnaa.infn.it)
- [indigo-aai.slack.com](https://indigo-aai.slack.com)

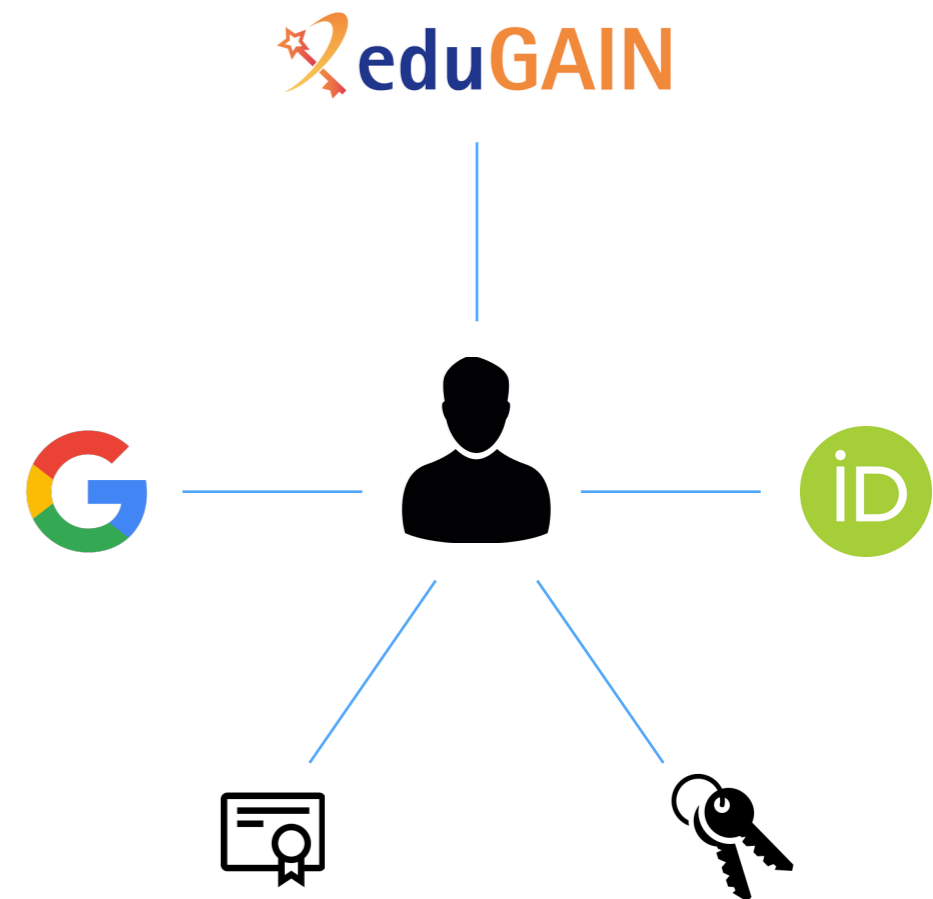
# Flexible authentication & account linking

Authentication supported via

- **local username/password** credentials (created at registration time)
- **SAML** Home institution IdP (e.g., EduGAIN)
- **OpenID Connect** (Google, Microsoft, Paypal, ORCID)
- **X.509** certificates

Users can link any of the supported authentication credentials to their IAM account at registration time or later

To link an external credential/account, the user has to **prove** that he/she owns such account



# User enrollment & registration service

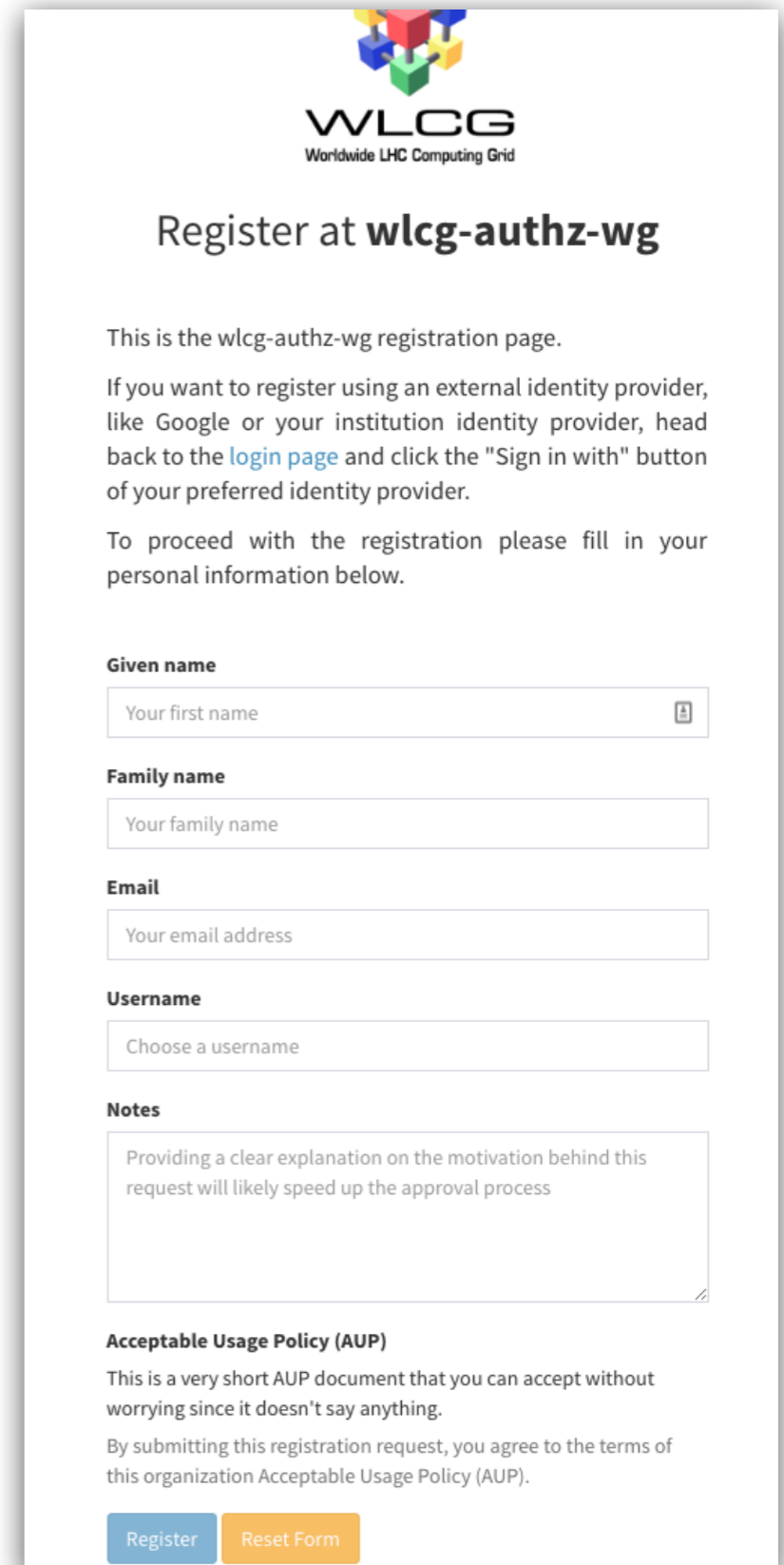
IAM supports two **enrollment flows**:

## Admin-moderated flow

- The applicant fills basic registration information, accepts AUP, proves email ownership
- VO administrators are informed by email and can approve or reject incoming membership requests
- The applicant is informed via email of the administrator decision

## Automatic-enrollment flow

- Users authenticated at **trusted**, **configurable** SAML IdPs are automatically on-boarded, without administrator approval



The screenshot shows the registration page for WLCG (Worldwide LHC Computing Grid) at wlcg-authz-wg. The page includes the WLCG logo, a title 'Register at wlcg-authz-wg', and introductory text. It contains several form fields: 'Given name' (with a placeholder 'Your first name'), 'Family name' (with a placeholder 'Your family name'), 'Email' (with a placeholder 'Your email address'), and 'Username' (with a placeholder 'Choose a username'). There is also a 'Notes' section with a text area containing the text 'Providing a clear explanation on the motivation behind this request will likely speed up the approval process'. Below the form is an 'Acceptable Usage Policy (AUP)' section with a short document and a note that by submitting the request, the user agrees to the terms. At the bottom, there are two buttons: 'Register' (blue) and 'Reset Form' (orange).

# User enrollment & registration service

IAM supports two **enrollment flows**:

## Admin-moderated flow

- The applicant fills basic registration information, accepts AUP, proves email ownership
- VO administrators are informed by email and can approve or reject incoming membership requests
- The applicant is informed via email of the administrator decision

## Automatic-enrollment flow

- Users authenticated at **trusted**, **configurable** SAML IdPs are automatically on-boarded, without administrator approval

The screenshot displays the WLCG registration service interface. At the top, the WLCG logo and 'Worldwide LHC Computing Grid' are visible. The user 'Andrea Ceccanti' is logged in. The main section is titled 'Requests' and contains a search bar and pagination controls. A table lists a registration request for Carlos Armando Garcia, created 8 hours ago. The request is in a 'CONFIRMED' status. Below the table, a detailed view of the request is shown, including the user's name, username (charlos1204), email (carlos.garcia@helmholtz-muenchen.de), and notes. At the bottom, there is a warning about the Acceptable Usage Policy (AUP) and buttons for 'Register' and 'Reset Form'.

Created	User	Request	Actions
8 hours ago	Carlos Armando Garcia	Registration request	<a href="#">Approve</a> <a href="#">Reject</a>

<b>Created</b>	07/06/2018 09:17:33
<b>Current Status</b>	CONFIRMED
<b>Name</b>	Carlos Armando Garcia
<b>Username</b>	charlos1204
<b>E-mail</b>	carlos.garcia@helmholtz-muenchen.de
<b>Notes</b>	I will attend the "Data Science - Curso 2018-19 - Santander - Peninsula de la Magdalena"

worrying since it doesn't say anything.  
By submitting this registration request, you agree to the terms of this organization Acceptable Usage Policy (AUP).

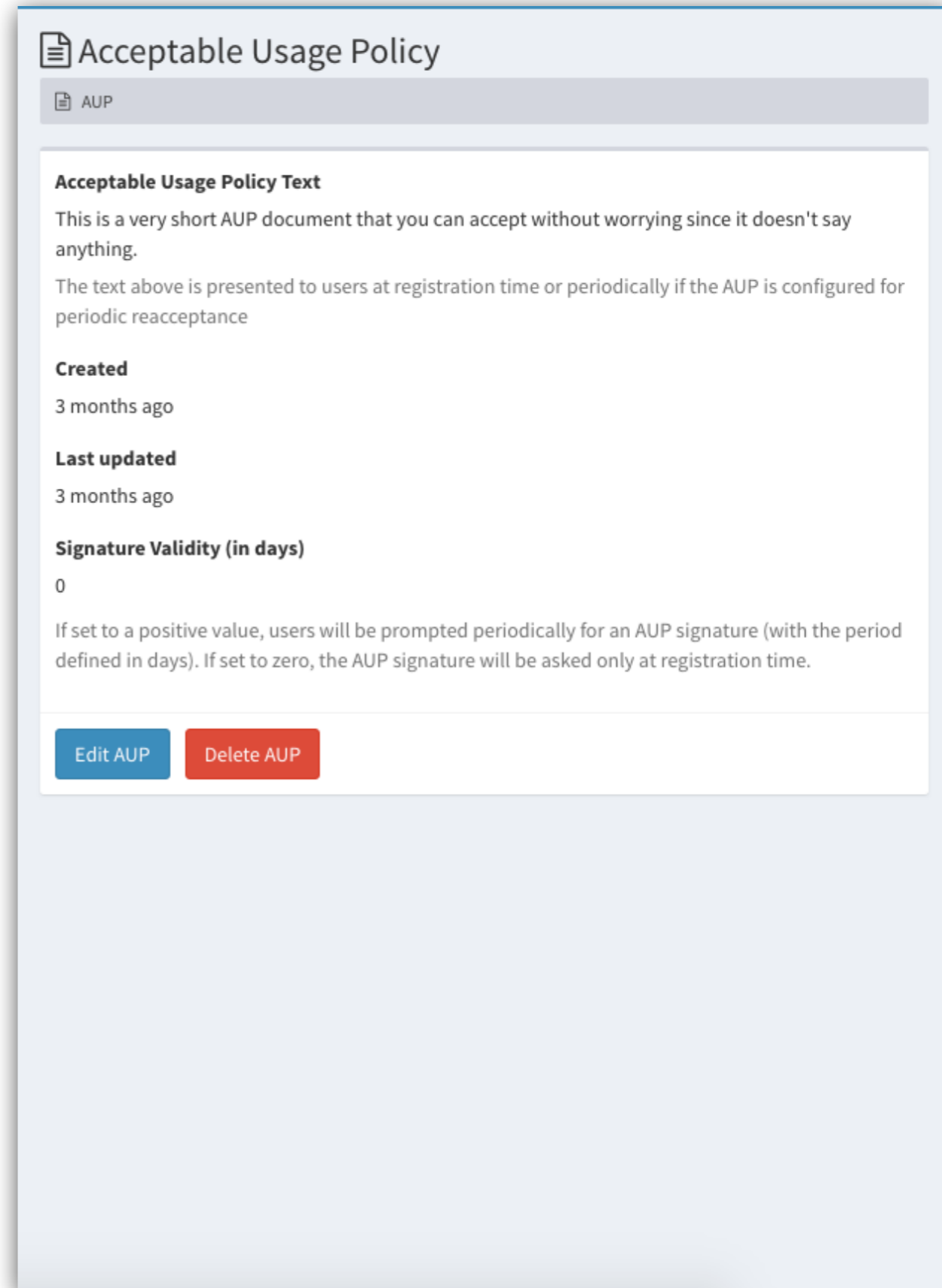
[Register](#) [Reset Form](#)

# AUP enforcement support

**AUP acceptance**, if enabled, can be configured to be:

- requested once at user registration time
- periodically, with configurable period

User cannot login to the system (and as such be authenticated at authorized at services) unless the **AUP** has been accepted



The screenshot shows a web interface for configuring an Acceptable Usage Policy (AUP). The title is "Acceptable Usage Policy" with a document icon. Below the title is a tab labeled "AUP". The main content area is titled "Acceptable Usage Policy Text" and contains the following text: "This is a very short AUP document that you can accept without worrying since it doesn't say anything. The text above is presented to users at registration time or periodically if the AUP is configured for periodic reacceptance". Below the text are two fields: "Created" with the value "3 months ago" and "Last updated" with the value "3 months ago". There is also a field for "Signature Validity (in days)" with the value "0". A note below this field states: "If set to a positive value, users will be prompted periodically for an AUP signature (with the period defined in days). If set to zero, the AUP signature will be asked only at registration time." At the bottom of the configuration area are two buttons: "Edit AUP" (blue) and "Delete AUP" (red).

# IAM deployment strategies

IAM is a **Spring Boot** application

- currently based on the MitreID Connect
- deployed behind an **NGINX**
- stores data in a **MariaDB/ MySQL** database

**Horizontally scalable**

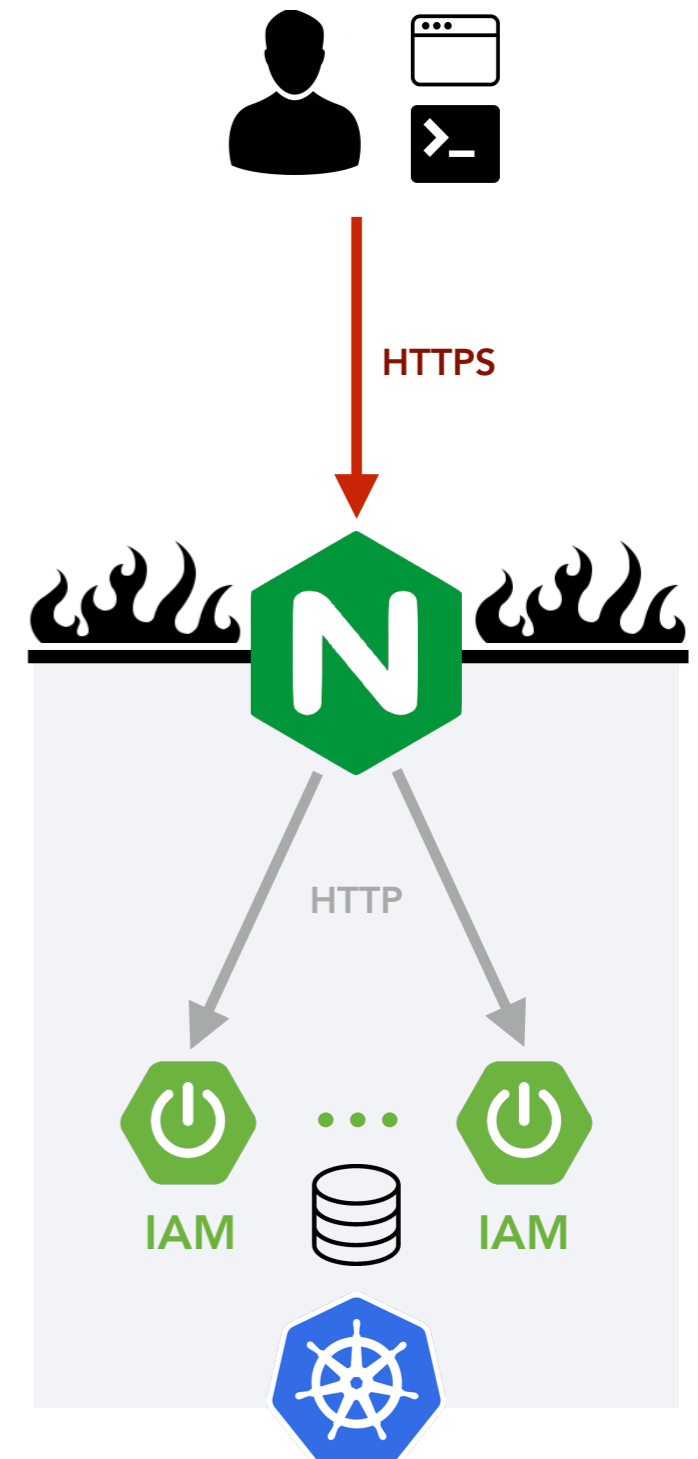
- all state persisted in the database

We deploy IAM as a **containerized** service on top of **Kubernetes**

- autoscaling, zero downtime rolling updates

Packages available for

- CENTOS 7, UBUNTU 1604



# IAM Software Quality

Aim to have **>90% unit test coverage on all code:**

- now 24k LoC, 85.6% branch coverage, >800 tests

**Open, test-driven** development process

**Static analysis** tools

- [SonarCube IAM page](#)

**Multiple test suites**

- **Unit tests**
- **Frontend test suite** (based on Selenium and Robot framework)
- **Deployment tests** (in CI)

The screenshot displays a GitHub pull request interface. At the top, there are three summary cards: 'Coverage' showing 85.6% (818 Unit Tests) and 'Coverage on New Code' at 0%; 'Duplications' showing 3.8% (72 Duplicated Blocks) and 'Duplications' at +0.0%; and 'Size' showing 0.4k. Below these is the pull request title 'Add support to multiple OIDC providers #249' by marcocaberletti. The interface includes tabs for Conversation (1), Commits (2), Checks (0), and Files changed (35). A timeline of activity shows the PR was created, reviewed, and added to the 'IAM next release' branch. A SonarQube bot comment reports one issue: 'OidcConfiguration.java#L97: Method has 10 parameters, which is greater than 7 authorized.' A 'Review requested' banner is visible at the bottom.

# IAM evolution: porting to Keycloak

IAM 2 (in development) will be based on Keycloak

- Powerful RedHat SSO solution
- Vibrant community: > 250 GitHub contributors
- LDAP/Kerberos integration
- Multi-tenancy



We will **focus on what not already provided** by Keycloak

- flexible registration service
- X.509 and VOMS authentication support



**Improved flexibility and sustainability**