

# Authentication



*Katarina Valalikova*

*@KValalikova*

*k.valalikova@evolveum.com*

# Agenda

- ❖ History
- ❖ Multi-factor, adaptive authentication
- ❖ SSO, SAML, OAuth, OpenID Connect
- ❖ Federation

# Who am I?

- ❖ Ing. Katarina Valaliková
- ❖ FIIT STU BA
- ❖ Evolveum s.r.o.
- ❖ Java Developer
- ❖ Identity Engineer



**Back to the topic**

# Authentication

- ❖ Verification if the user is who he claims to be.

# Authentication

- ❖ Three main factors:
  - ❖ Something that I know - password, passphrase,...
  - ❖ Something that I have - token, certificate, ...
  - ❖ Something that I am - biometrics, ...

# Authentication

- ❖ The most usual use case
  - ❖ username & password

# Brief history





# Authentication

- ❖ First used in 60s in 20th century - MIT
- ❖ Username and password saved in plain text on the filesystem
- ❖ Hashing
- ❖ Multi-factor authentication
- ❖ Adaptive authentication

# Authentication

<Cthon98> hey, if you type in your pw, it will show as stars  
<Cthon98> \*\*\*\* see!  
<AzureDiamond> hunter2  
<AzureDiamond> doesnt look like stars to me  
<Cthon98> <AzureDiamond> \*\*\*\*  
<Cthon98> thats what I see  
<AzureDiamond> oh, really?  
<Cthon98> Absolutely  
<AzureDiamond> you can go hunter2 my hunter2-ing hunter2  
<AzureDiamond> haha, does that look funny to you?  
<Cthon98> lol, yes. See, when YOU type hunter2, it shows to us as \*\*\*\*  
<AzureDiamond> thats neat, I didnt know IRC did that  
<Cthon98> yep, no matter how many times you type hunter2, it will show to us as \*\*\*\*  
<AzureDiamond> awesome!  
<AzureDiamond> wait, how do you know my pw?  
<Cthon98> er, I just copy pasted YOUR \*\*\*\*'s and it appears to YOU as hunter2 cause its your pw  
<AzureDiamond> oh, ok.

# Multi-factor authentication

- ❖ Combination of more than 1 factors
  - ❖ something that I know
  - ❖ something that I have
  - ❖ something that I am

# Adaptive authentication

- ❖ Risk-factor authentication
- ❖ Deciding if the user needs to use additional authentication
- ❖ Example: internet banking

# Adaptive authentication

- ❖ Device recognition

- ❖ Do I log in with the known device?

# Adaptive authentication

- ❖ Device recognition
- ❖ Threat services
  - ❖ IP Whitelist
  - ❖ IP Blacklist

# Adaptive authentication

- ❖ Device recognition
- ❖ Threat services
- ❖ **Directory lookup**
  - ❖ Checking user's standard profile against known directory

# Adaptive authentication

- ❖ Device recognition
- ❖ Threat services
- ❖ Dictionary lookup
- ❖ Geo-location
  - ❖ Checking typical user location where he logs in from



# Adaptive authentication

- ❖ Device recognition
- ❖ Threat service
- ❖ Dictionary lookup
- ❖ Geo-location
- ❖ Geo-velocity
  - ❖ If I log in from Kosice, I won't log in from USA in 10mins.

# Adaptive authentication

- ❖ Device recognition
- ❖ Threat service
- ❖ Dictionary lookup
- ❖ Geo-location
- ❖ Geo-velocity
- ❖ **Geo-Fencing**
  - ❖ Geographic barriers

# Adaptive authentication

- ❖ Device recognition
- ❖ Threat service
- ❖ Dictionary lookup
- ❖ Geo-location
- ❖ Geo-velocity
- ❖ Geo-fencing
- ❖ Behavioural biometrics
  - ❖ Suspicious user's keystroke and mouse movements

# Adaptive authentication

- ❖ Device recognition
- ❖ Threat services
- ❖ Dictionary lookup
- ❖ Geo-location
- ❖ Geo-velocity
- ❖ Geo-fencing
- ❖ Behavioural biometrics
- ❖ Identity Governance
  - ❖ Decisions made according to the user's access rights - risk factor

# Adaptive authentication

- ❖ Device recognition
- ❖ Threat service
- ❖ Dictionary lookup
- ❖ Geo-location
- ❖ Geo-velocity
- ❖ Geo-fencing
- ❖ Behavioural biometrics
- ❖ Identity Governance
- ❖ User behaviour analytics
  - ❖ How does the user usually behave?

# HTTP Authentication (rfc2617)

- ❖ simple challenge-response authentication mechanism
- ❖ used by a server to challenge a client request
- ❖ used by a client to provide authentication information

# HTTP Authentication (rfc2617)

- ❖ extensible, case-insensitive token to identify authentication scheme
- ❖ comma-separated list of attribute-value pairs carrying the parameters necessary for achieving authentication

# HTTP Authentication (rfc2617)

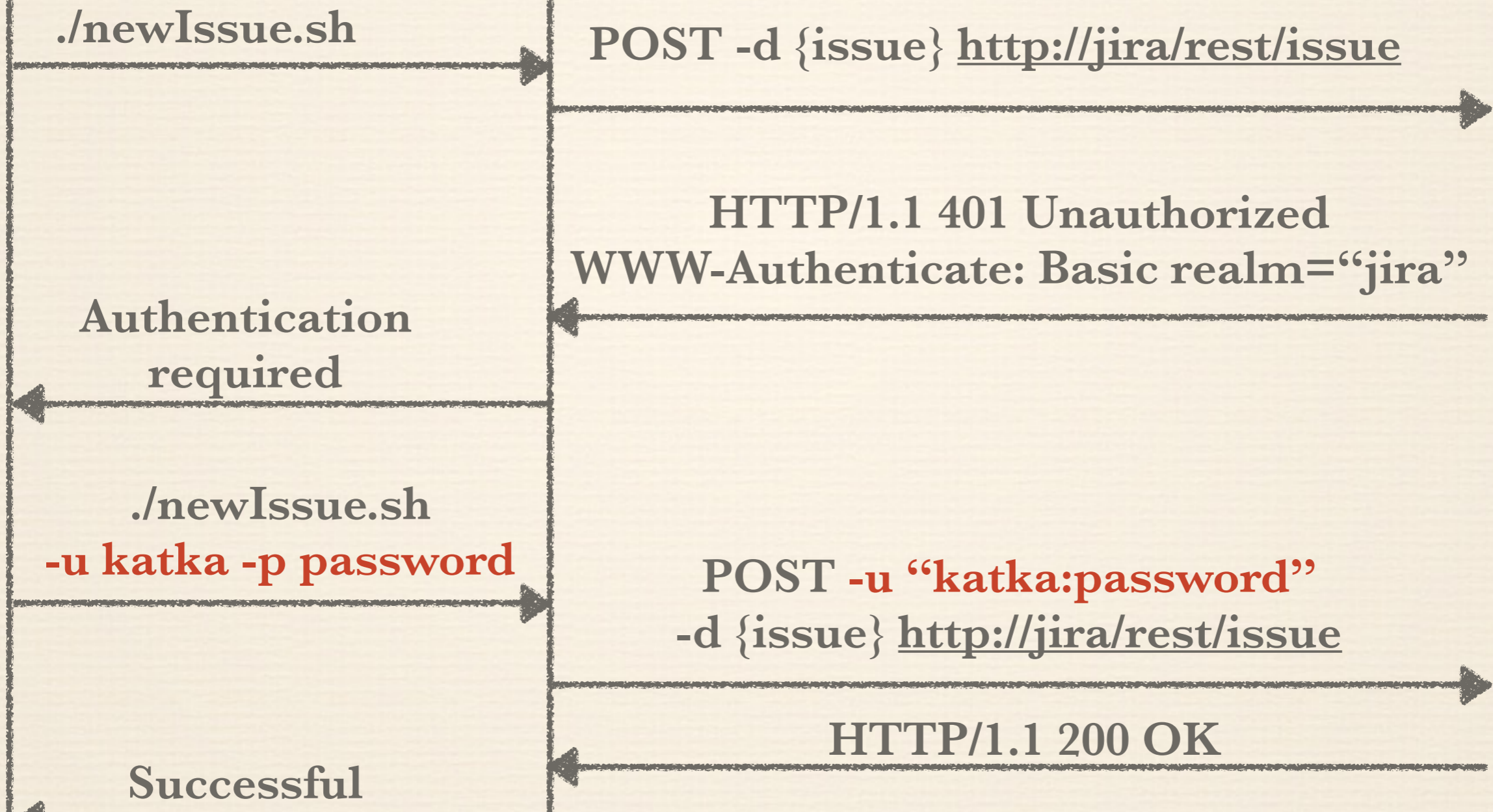
- ❖ 401 (Unauthorized) response is used by a server to challenge the authorization of a user agent
- ❖ Response must include WWW-Authenticate header containing at least one applicable challenge



# HTTP Authentication (rfc2617)

- ❖ Responsibility for parsing WWW-Authentication is left for the user agents, because
  - ❖ WWW-Authentication header can contain more than one challenge
  - ❖ Challenge can contain comma-separated list of authentication parameters

# HTTP Authentication (rfc2617)



# Single Sign-On

- ❖ Unified log in process for different applications
- ❖ Log in once and share it with others
- ❖ Just one username and password

# Single Sign-On

- ❖ Service provider - Relying party
- ❖ end application which the user wants to access
- ❖ authentication is left to Identity Provider

# Single Sign-On

- ❖ Identity provider
- ❖ service providing authentication for user
- ❖ creates a session for logged in user

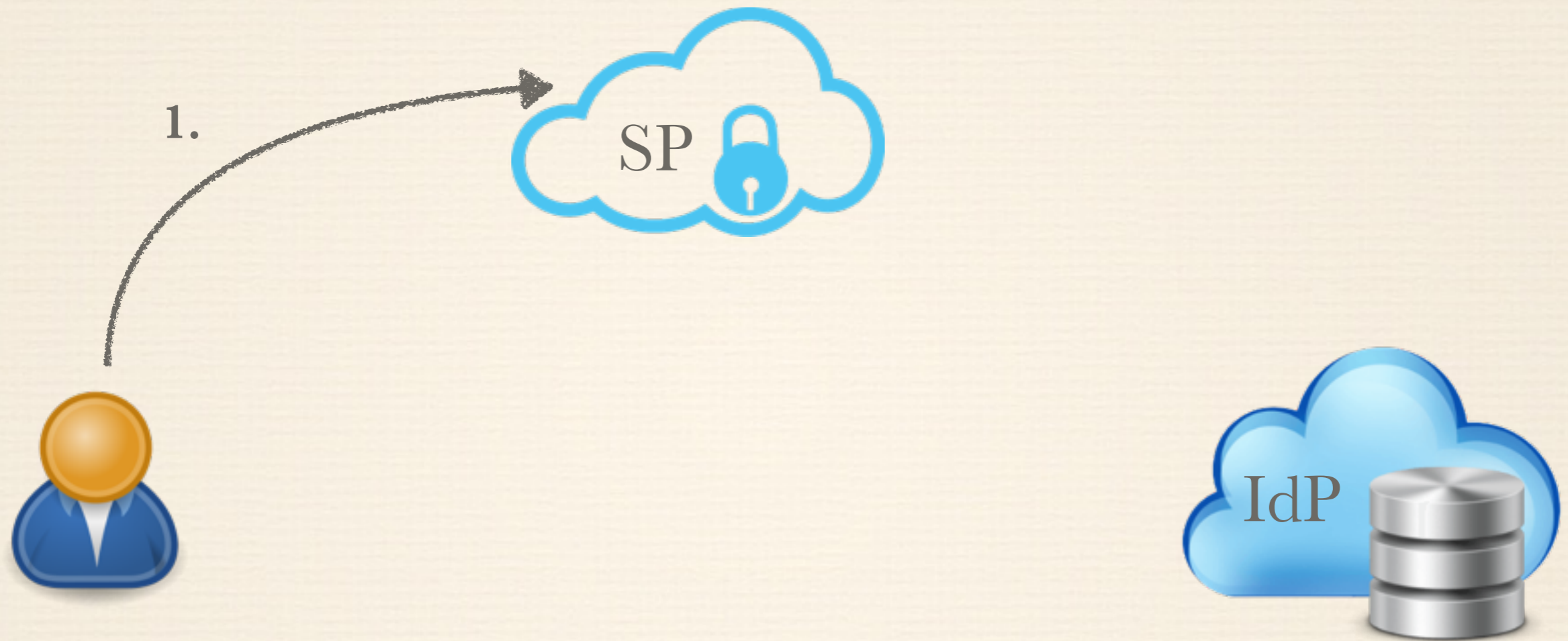
# Single Sign-On

- ❖ Session
  - ❖ information about logged in user for some period of time
  - ❖ logged-in user doesn't need to log in again

# Single Sign-On



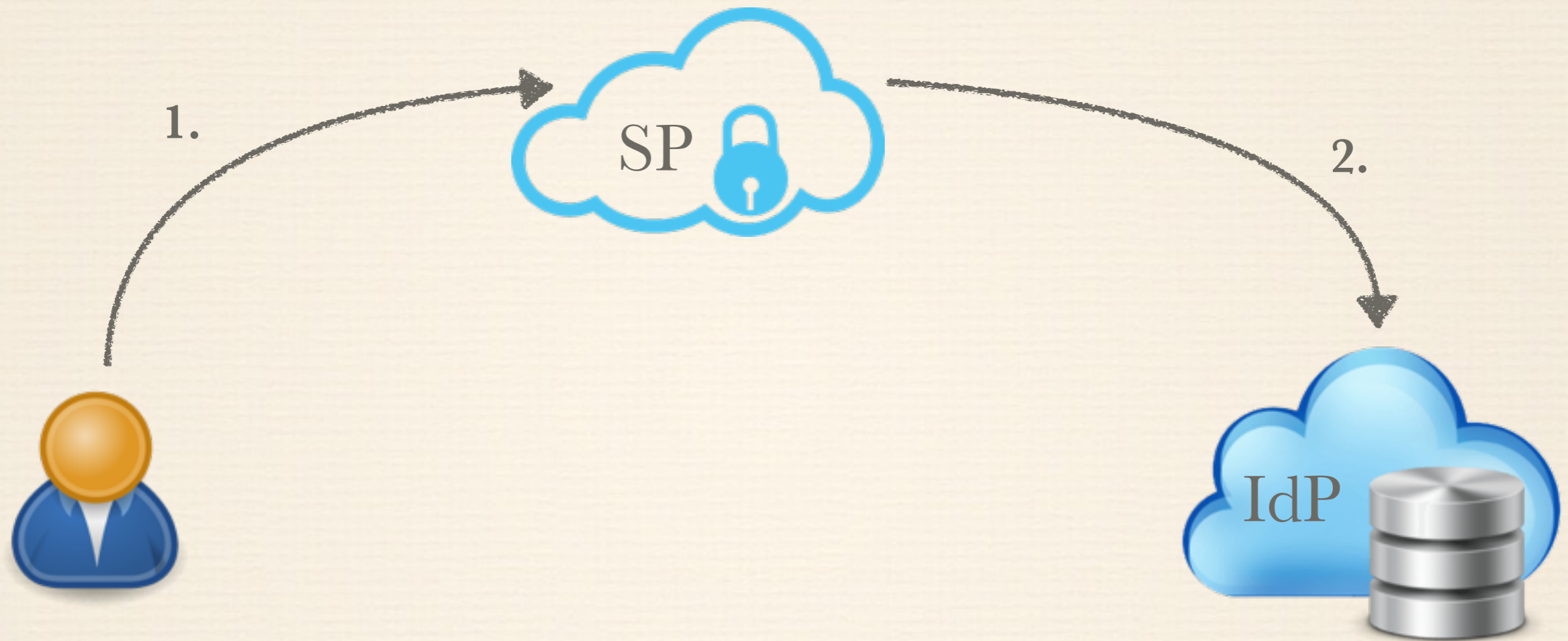
# Single Sign-On



1. User who hasn't been authenticated yet wants to use the application (SP - Service Provider)

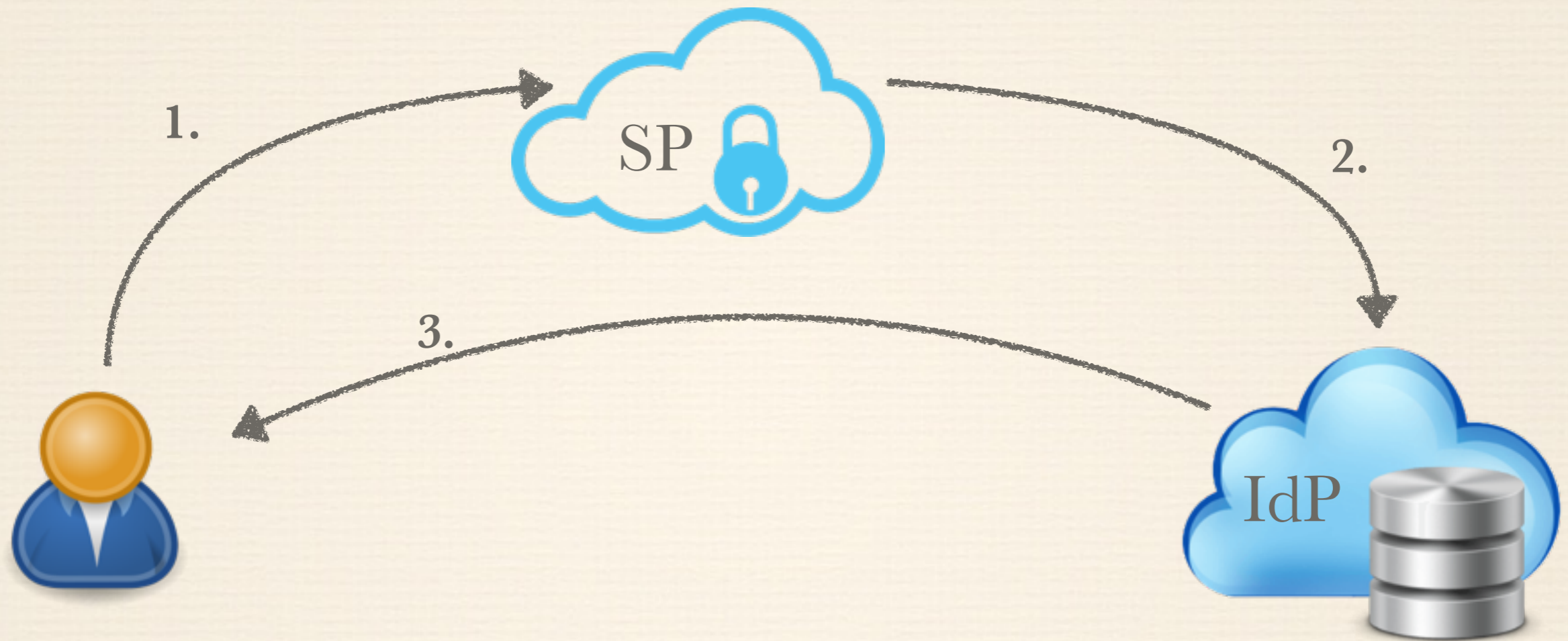


# Single Sign-On



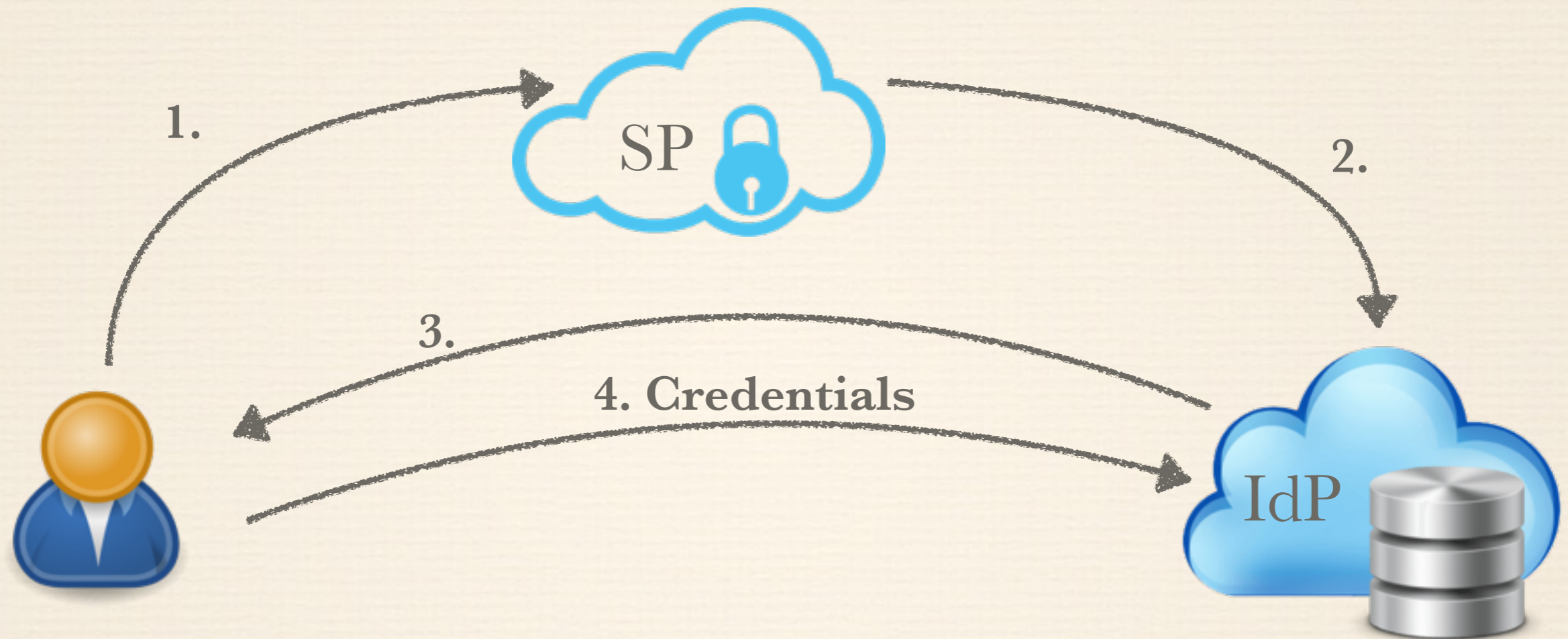
2. Application which is a part of SSO solution verifies from IdP if the user was authenticated before.

# Single Sign-On



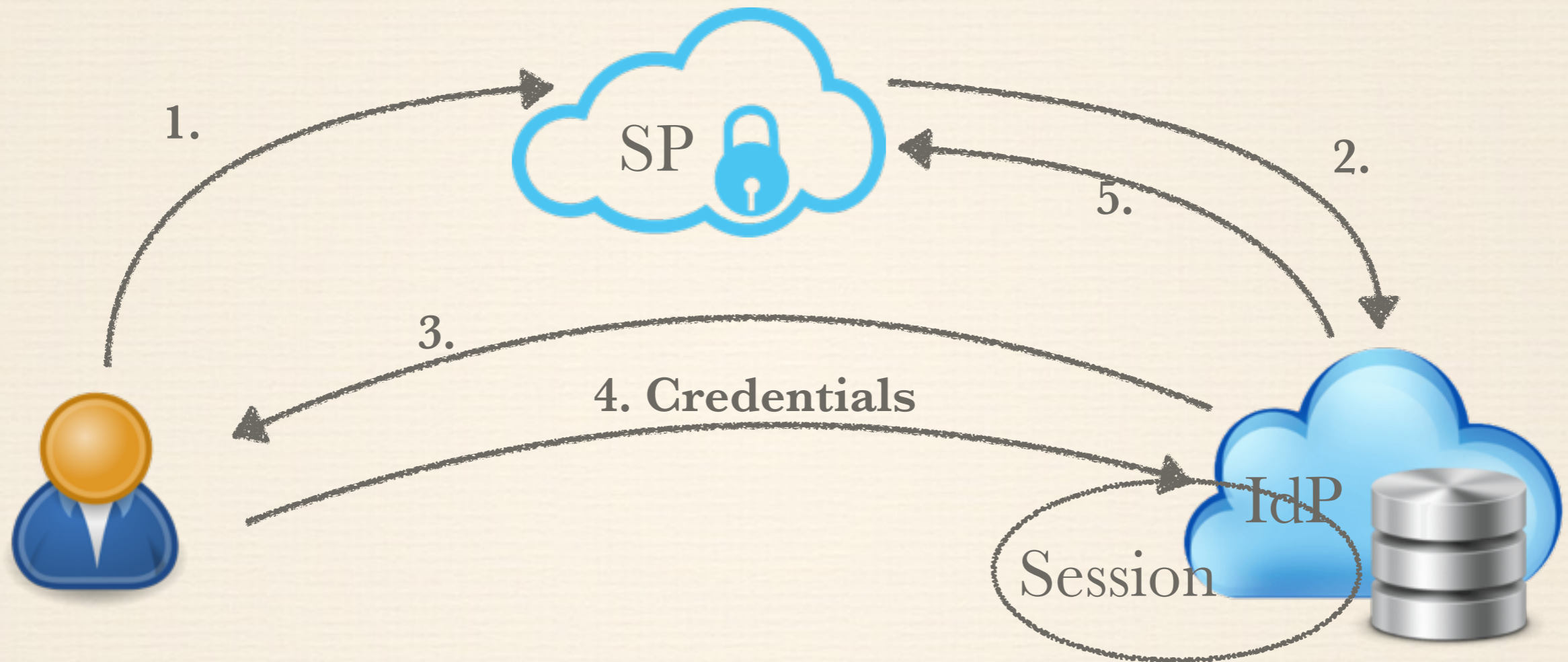
3. IdP found out that the user hasn't been logged in yet.  
User is redirected to the login page.

# Single Sign-On



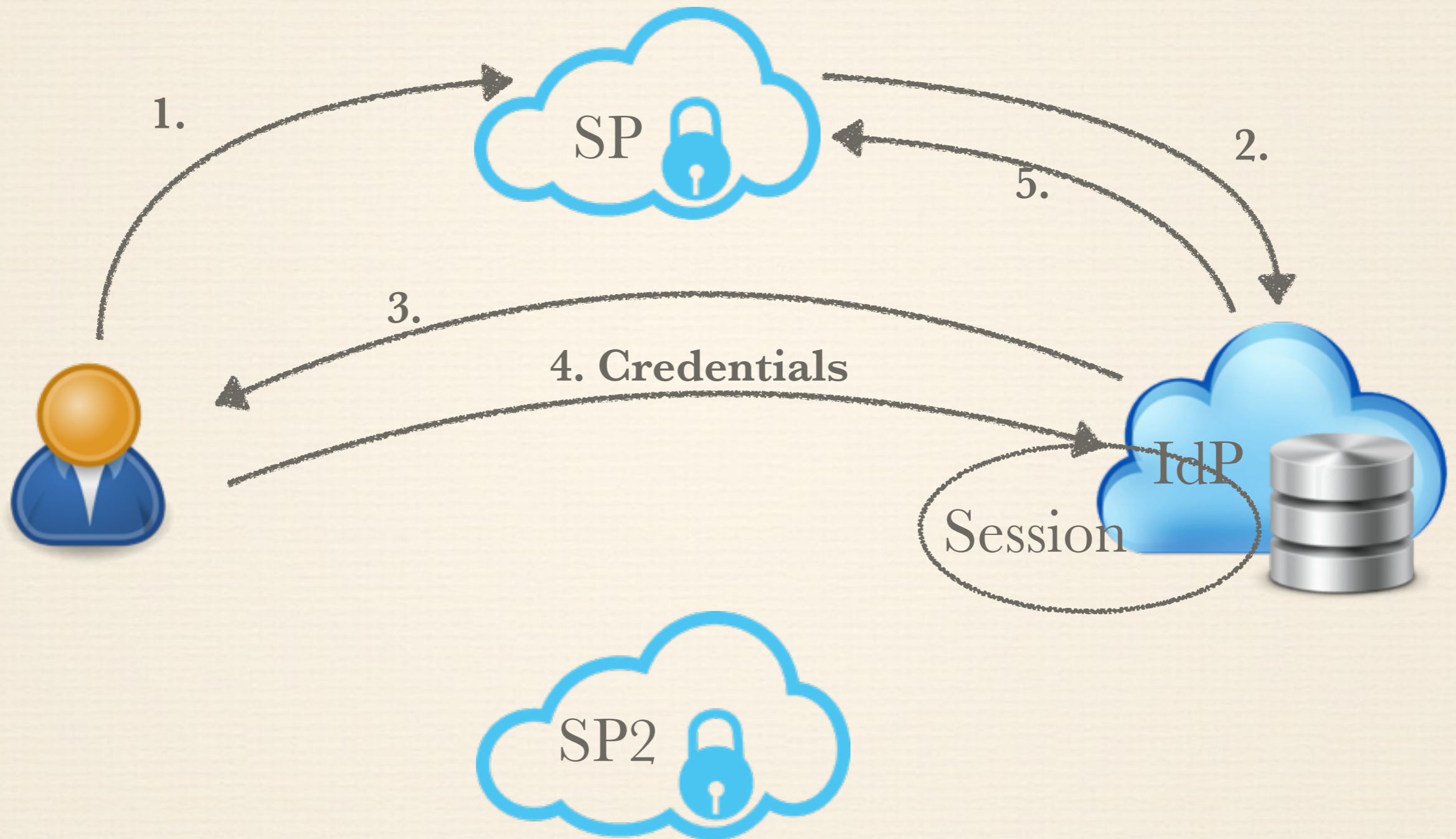
4. User fill in his credentials and submit the login form.

# Single Sign-On

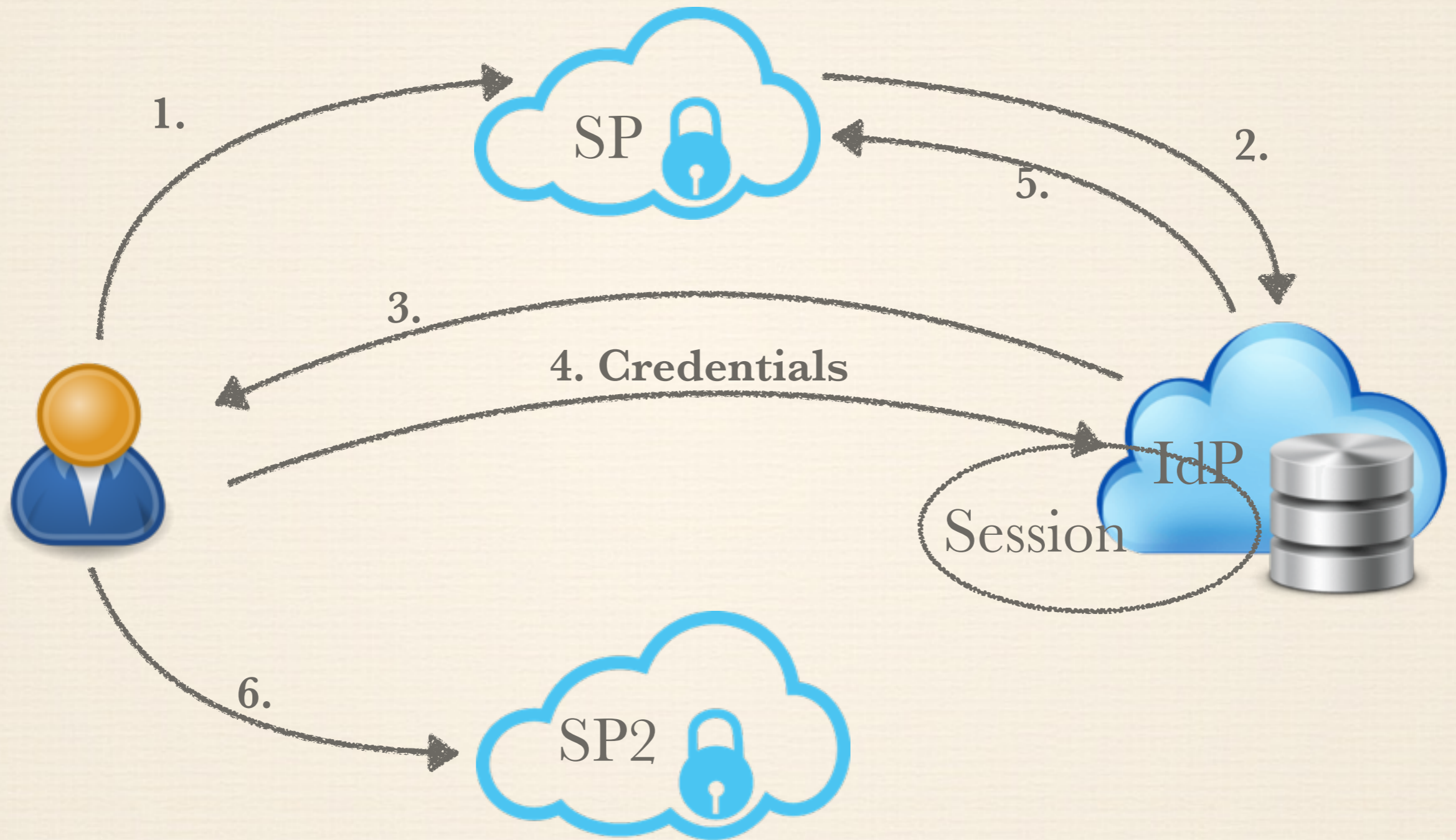


5. IdP successfully authenticates user. User is redirected to the origin application (SP). IdP creates a SSO session for the user.

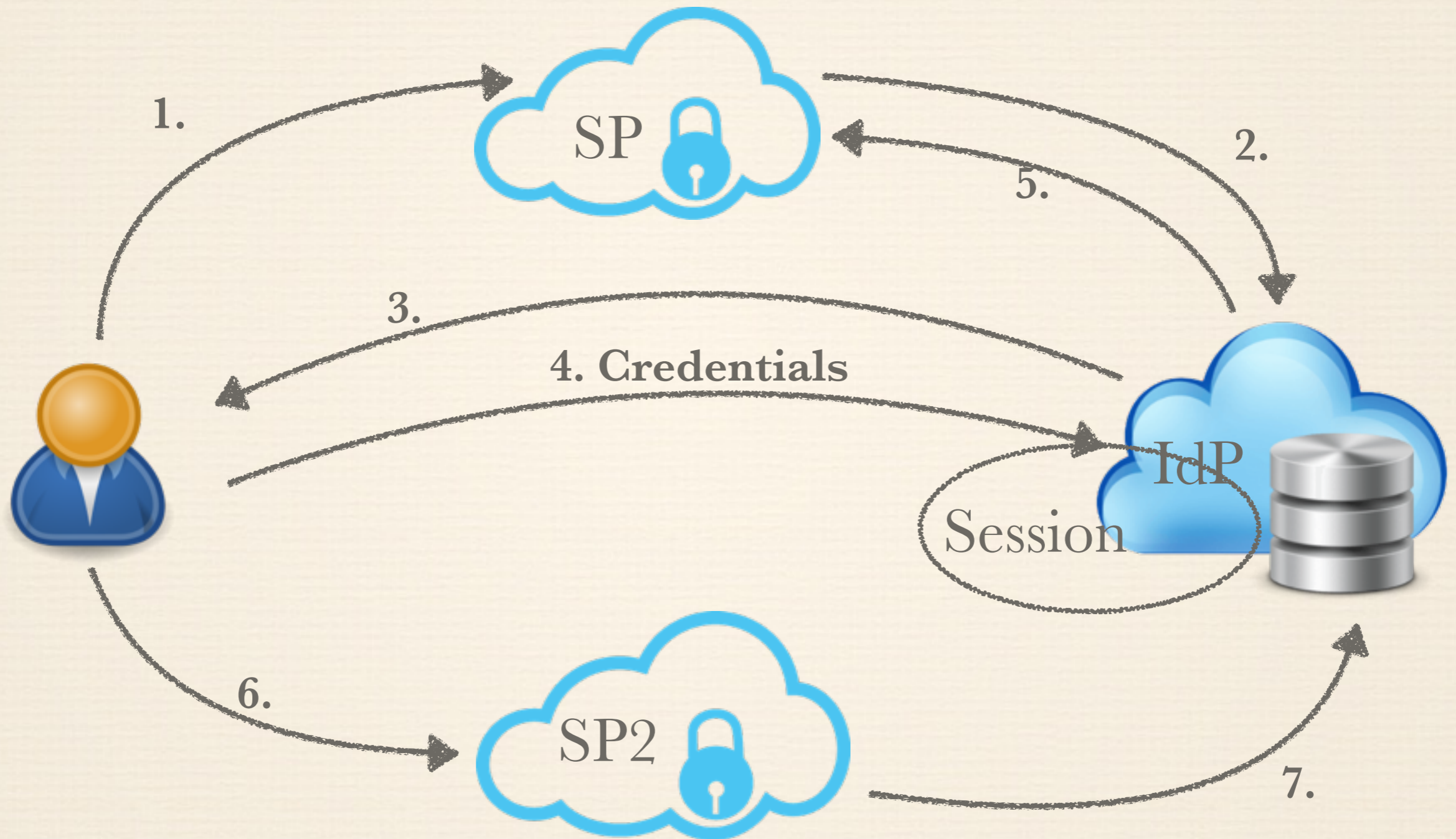
# Single Sign-On



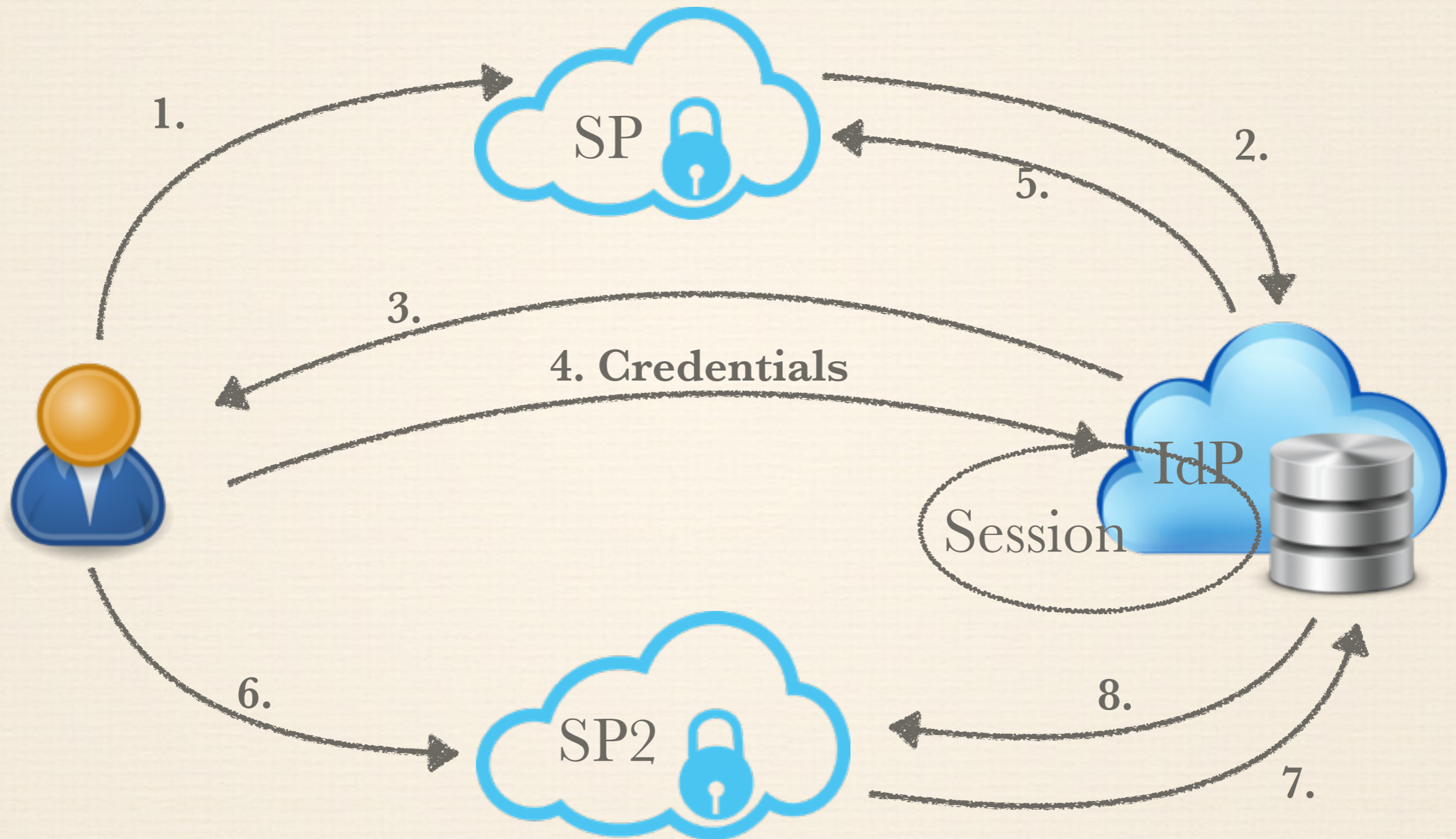
# Single Sign-On



# Single Sign-On



# Single Sign-On





# SAML

- ❖ Security Assertion Markup Language
- ❖ XML-based framework for describing and exchanging security information between online partners
- ❖ Expressed in the form of portable SAML assertions
- ❖ Applications across security domain boundaries can trust
- ❖ OASIS SAML standard defines precise syntax and rules for:
  - ❖ requesting
  - ❖ creating
  - ❖ communicating
  - ❖ using SAML assertions

# SAML

- ❖ Different drives behind of adoption of the SAML standard:
  - ❖ Web SSO - standard vendor-independent grammar and protocol for transferring information about a user from one web server to another
  - ❖ Federated identity - to agree on and establish a common, shared name identifier to refer to the user in order to share information about the user across the organizational boundaries.
  - ❖ Web services and other industry standards - profile for how to use SAML's rich assertion constructs within a WS-Security security token that can be used, for example, to secure web service SOAP message exchanges

# SAML Participants

- ❖ SAML Asserting party
  - ❖ System entity making SAML assertion
  - ❖ Sometimes called SAML authority
- ❖ SAML Relying party
  - ❖ System entity using received assertion

# SAML

- ❖ System entities can operate in variety of SAML roles
  - ❖ define SAML services and protocol messages they will use
  - ❖ defined assertions they will generate or consume
  - ❖ for SSO - Identity Provider, Service Provider

# SAML Assertion

- ❖ Heart of the SAML assertion is subject
  - ❖ user, computer, organization to be authenticated
  - ❖ referred also as a principal

# SAML

## Profiles

Combinations of assertions, protocols, and bindings to support a defined use case

## Bindings

Mappings of SAML protocols onto standard messaging and communication protocols

## Protocols

Requests and responses for obtaining assertions and doing identity management

## Assertions

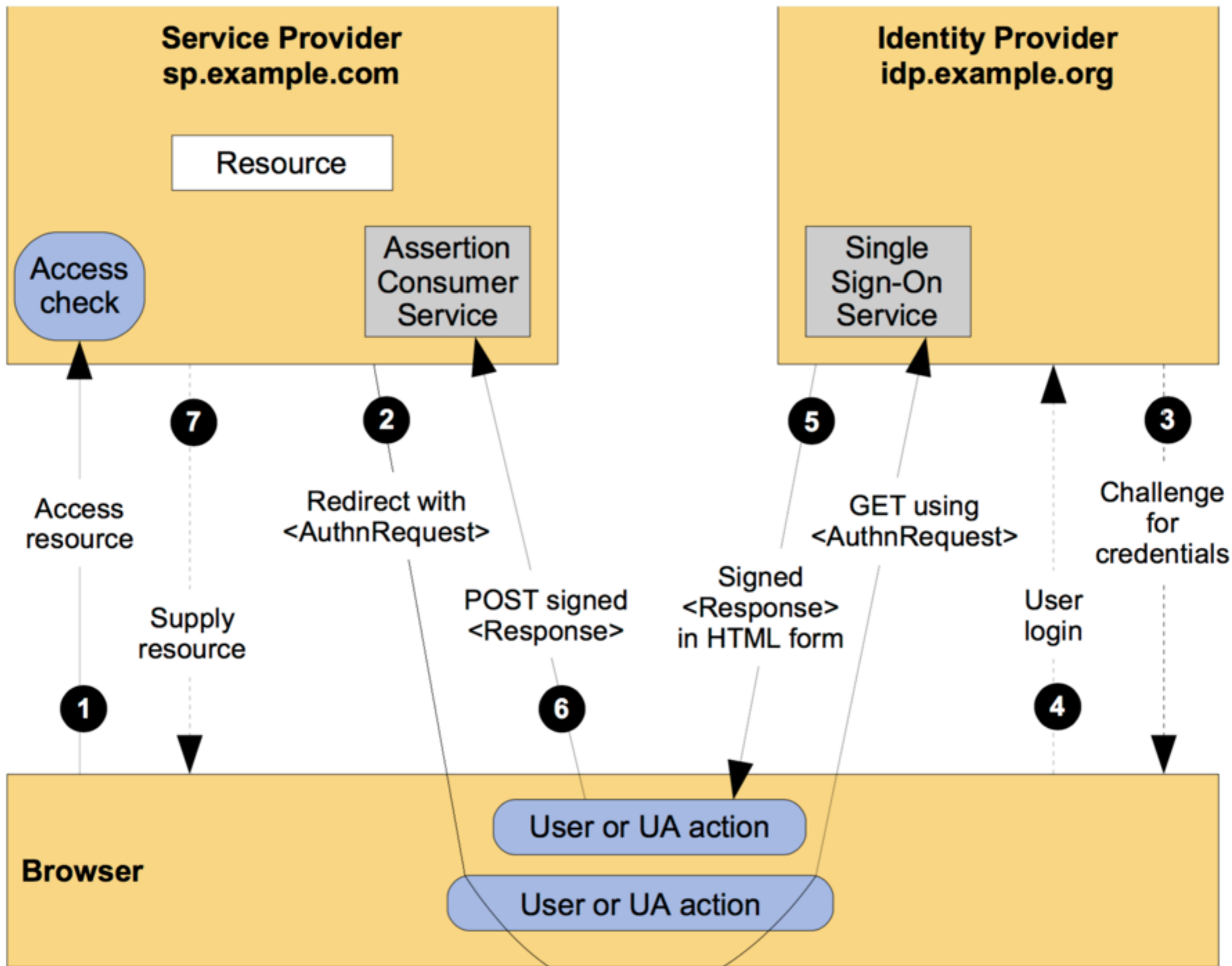
Authentication, attribute, and entitlement information

## Authentication Context

Detailed data on types and strengths of authentication

## Metadata

Configuration data for identity and service providers



# OAuth 2.0

- ❖ Security protocol used to protect a large number of web APIs
- ❖ Used to connect websites to one another
- ❖ Powers native and mobile applications connecting to cloud services



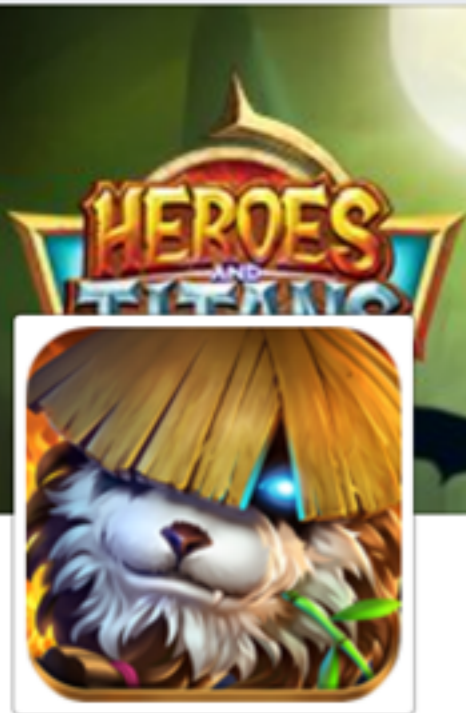
# OAuth 2.0

- ❖ Delegation protocol
- ❖ “a means of letting someone who controls a resource allow a software application to access that resource on their behalf without impersonating them”

# OAuth 2.0

❖ as specified in RFC 6749:

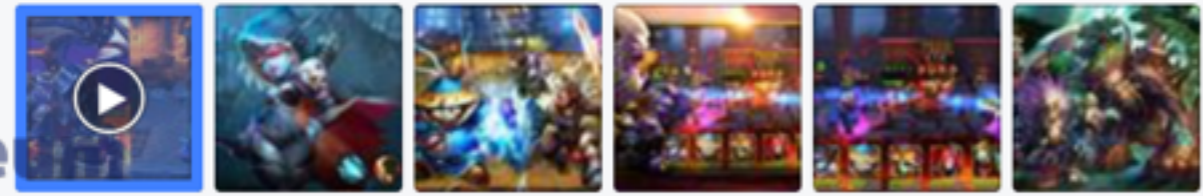
“The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.”



## Heroes and Titans

★★★★★

Role Playing Game · 50,000 players



[Play Now](#)

By clicking on "Play Now" above, Heroes and Titans will receive the following info: your public profile. ⓘ

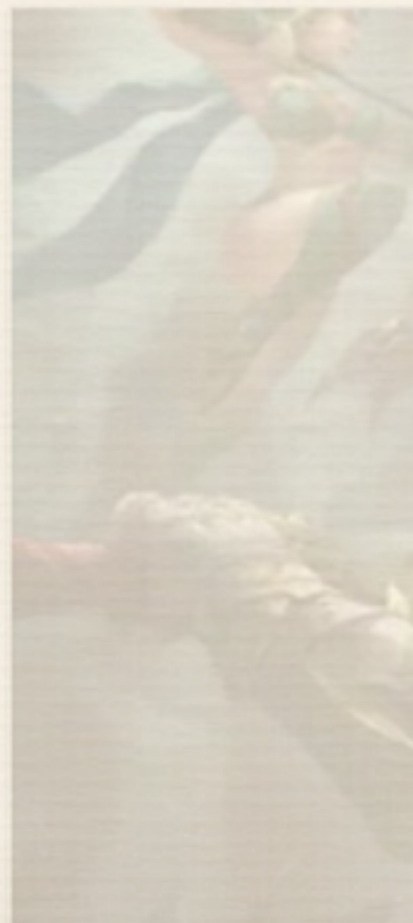
[Review the info you provide](#)

By proceeding, you agree to Heroes and Titans's [Terms of Service](#) and [Privacy Policy](#)


This doesn't let the app post to Facebook.

- [Share](#)
- [Block](#)
- [App Website](#)
- [Report a Problem](#)

Available on Facebook.com



## Play Now

By clicking on "Play Now" above, Heroes and Titans will receive the following info: your public profile. 

 Review the info you provide

By proceeding, you agree to Heroes and Titans's **Terms of Service** and **Privacy Policy**

 This doesn't let the app post to Facebook.

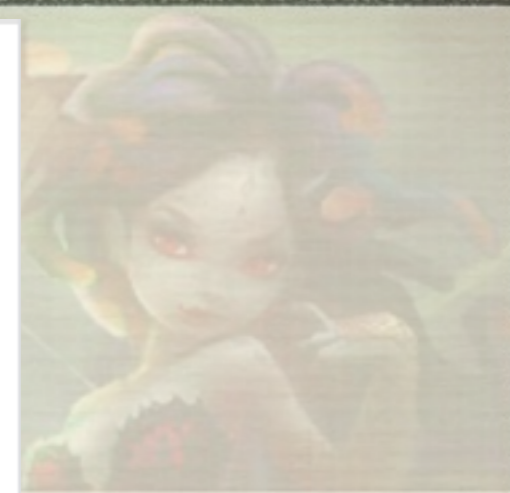
 Share

 Block

 App Website

 Report a Problem

Available on Facebook.com



Play Now

Now" above, Heroes and  
the following info: your public

ou provide

agree to Heroes and  
ervice and Privacy Policy

ne app post to Facebook.

n

ok.com

# OAuth 2.0

- ❖ Components
  - ❖ Resource owner - has access to the API and can delegate access to the API
  - ❖ Protected resource - component that the resource owner has access to.
  - ❖ Client - the piece of the software that accesses the protected resource on behalf of the resource owner
  - ❖ Authorization server - issues OAuth access tokens

# OAuth 2.0 - Obtaining authorization



Resource owner



User's agent



Authorization  
server



Client

# OAuth 2.0 - Obtaining authorization



Resource owner



User's agent

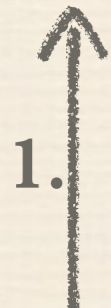
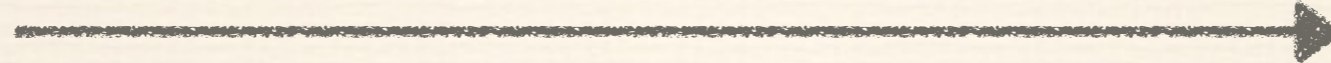


Authorization server

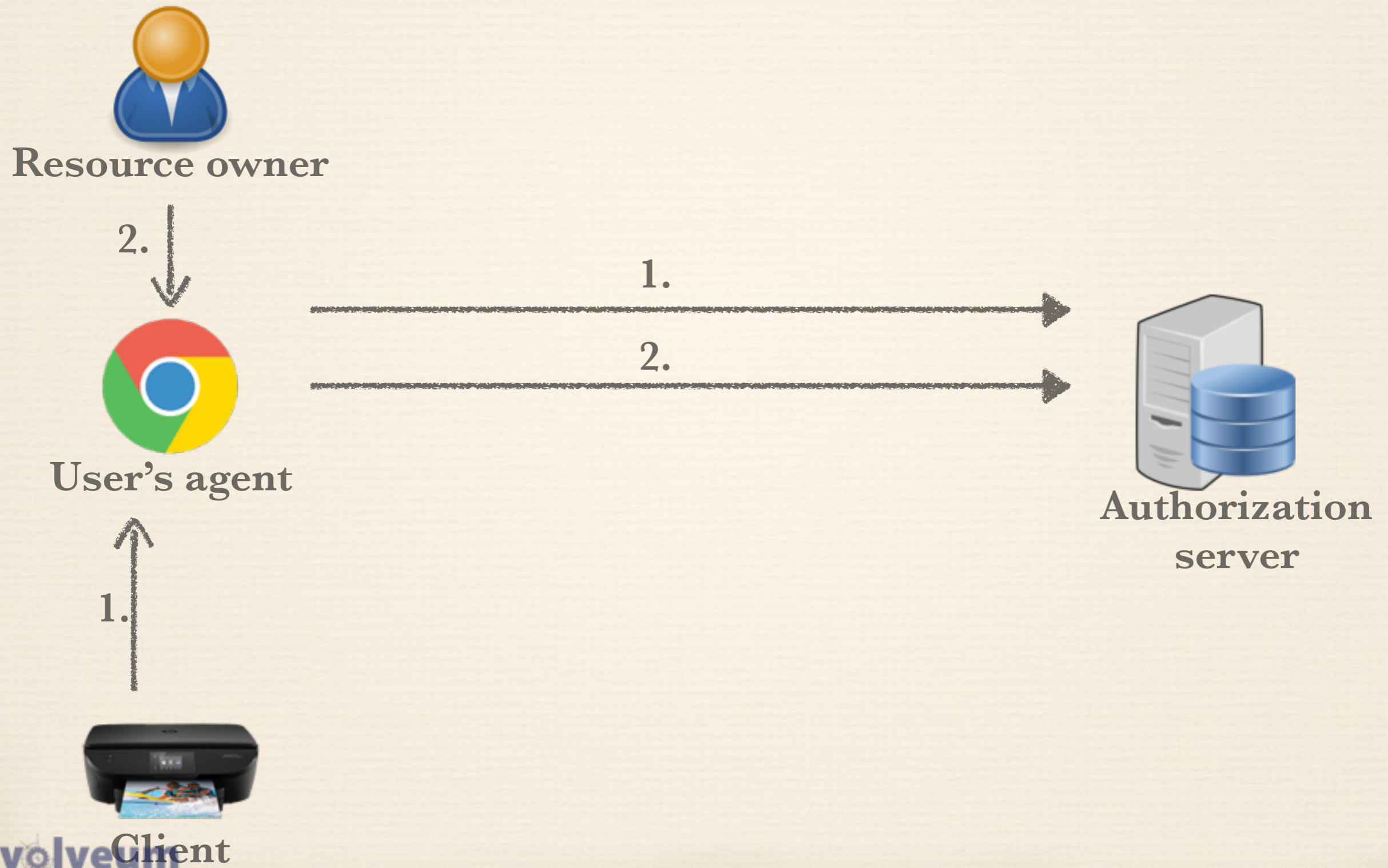


Client

1.

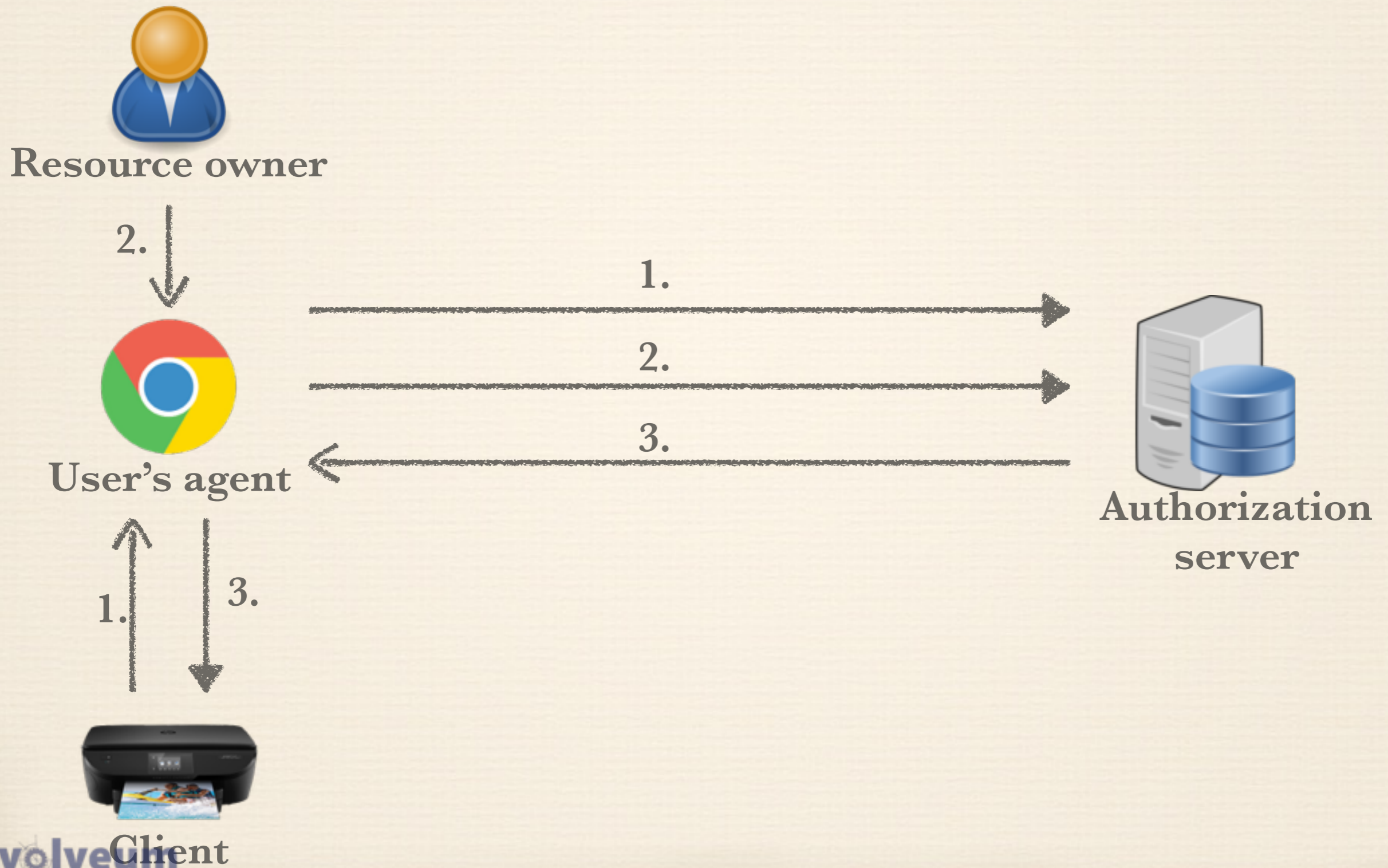


# OAuth 2.0 - Obtaining authorization

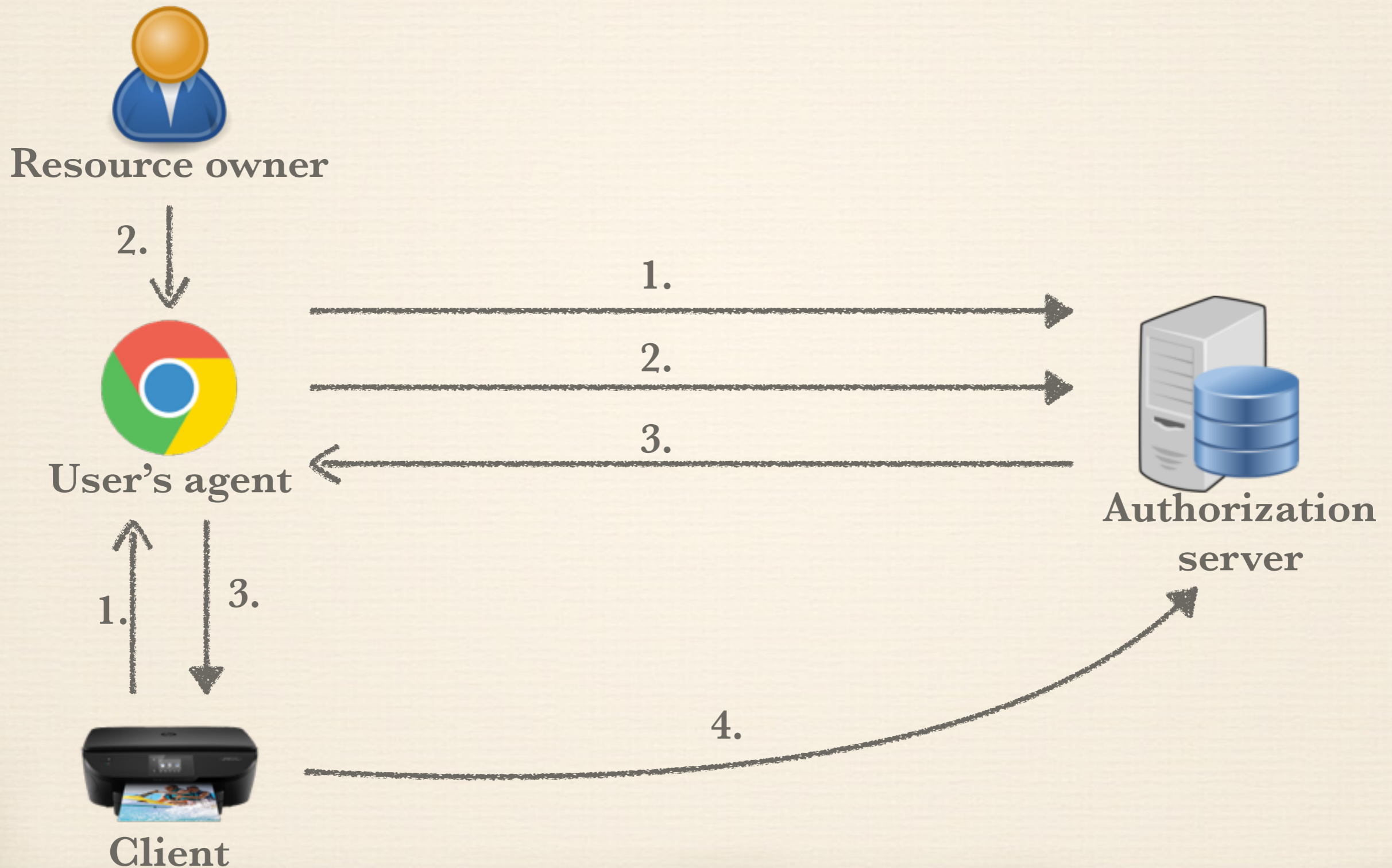




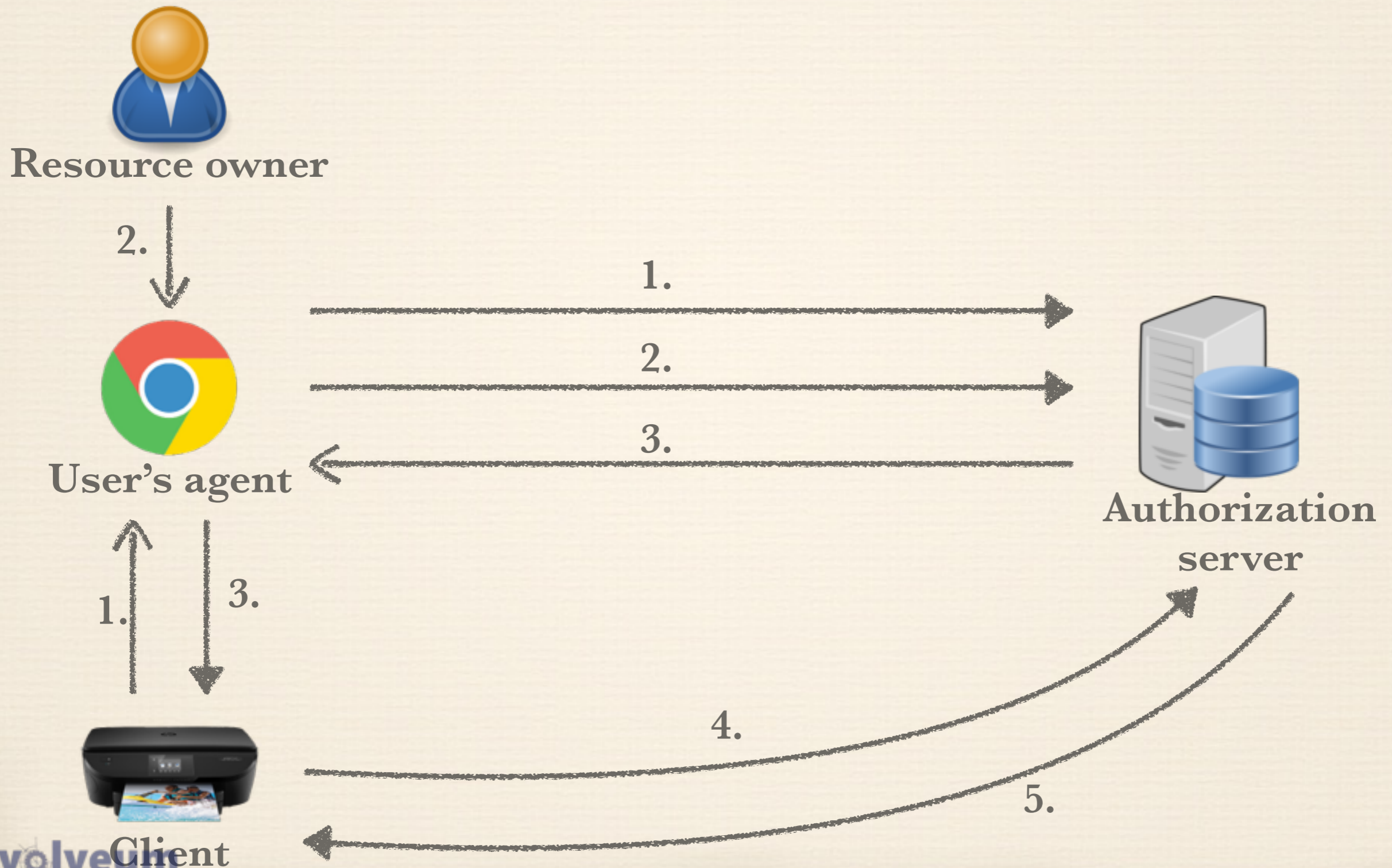
# OAuth 2.0 - Obtaining authorization



# OAuth 2.0 - Obtaining authorization



# OAuth 2.0 - Obtaining authorization



# Social login



**in** | Login with LinkedIn

**f** | Login with Facebook

or

Login with your SlideShare account

Keep me logged in

Login

[Forgot password?](#)

[Signup for a SlideShare account](#)

# OpenID Connect

- ❖ simple identity layer on top of the OAuth 2.0
- ❖ enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server
- ❖ enables Clients to obtain basic profile information about the End-User in an interoperable and REST-like manner
- ❖ OpenID Connect implements authentication as an extension to the OAuth 2.0 authorization process

# Federation



Thanks for the attention.

Any questions?

# Summary

- ❖ Authentication
- ❖ HTTP Authentication
- ❖ Multi-factor authentication
- ❖ Adaptive authentication
- ❖ SSO
- ❖ SAML
- ❖ OAuth 2.0
- ❖ OpenID Connect
- ❖ Federation



# Diplomové a bakalárske práce

- ❖ Smart audit log analysis
- ❖ Role Mining
- ❖ Visualisation of Identity Management system configuration
- ❖ Intelligent identity information processing
- ❖ Command line tool
- ❖ Mobile or self-service application