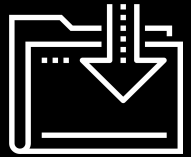


Course: Java
S1





Let's start with a simple question.



How comfortable are you giving out your login information to someone?

Even if it's someone (or some company) that you have complete trust in, what potential issues arise when you provide full access to everything you have secured behind a login?

Learning Outcomes

By the end of this lesson, the learner will be able to:

01

Define OAuth 2.

02

Describe federated authentication.

03

Describe delegated authorization.

04

Describe the role an Authorization Server plays in an OAuth 2 system.

05

Describe the role a Resource Server plays in an OAuth 2 system.

06

Define bearer token.

07

Define grant flow.

08

Describe the role clients play in an OAuth 2 system.

09

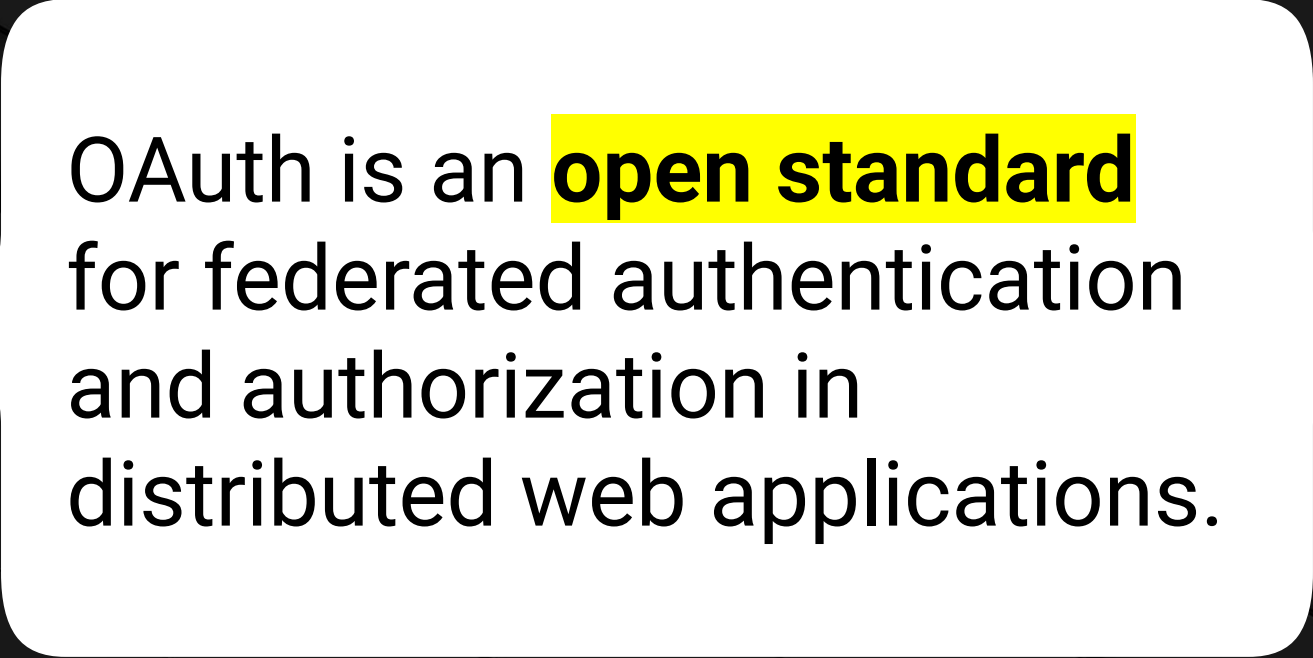
Describe the role client secrets play in an OAuth 2 system.



OAuth 2.0 Basics



What is OAuth?



OAuth is an **open standard**
for federated authentication
and authorization in
distributed web applications.

What is OAuth?



Sites such as Facebook and GitHub use this to allow their users to share account information with third parties.



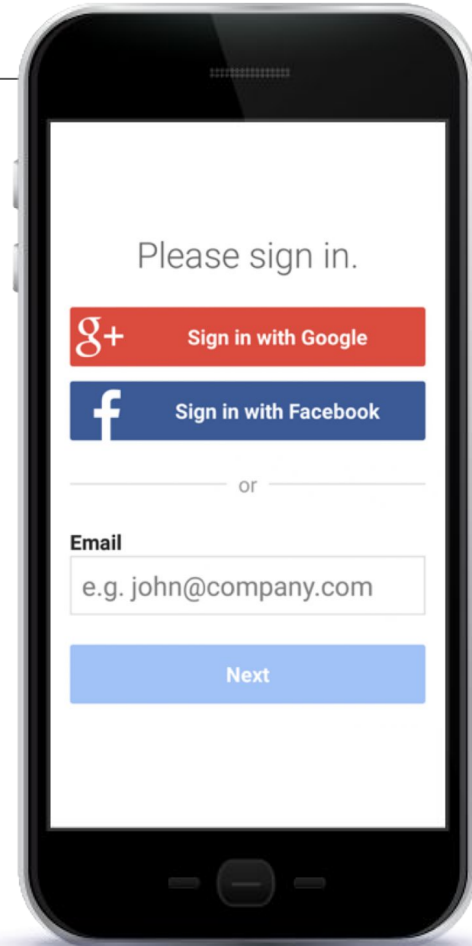
OAuth can also be used for single sign on and distributed access for cloud-native microservice based systems.

Federated Authentication

This means that one system delegates the responsibility of authentication to another system.

Enterprise systems do this by delegating to LDAP, SAML, or OpenID.

You have seen this with “Login with Facebook.” or “Log in with your Google account.” on various websites.



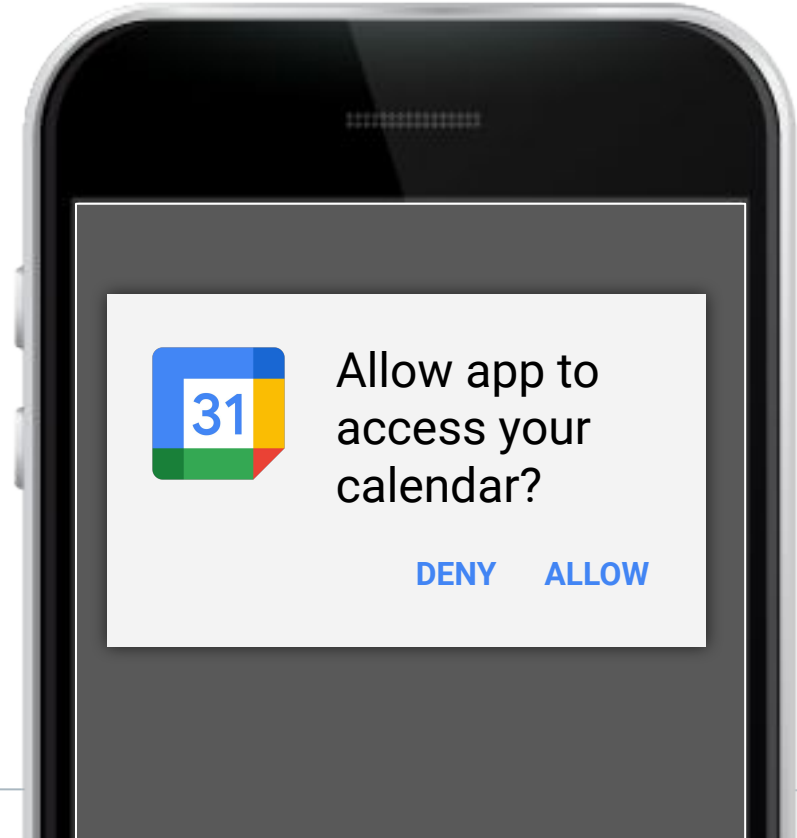
Delegated Authorization

This involves you giving another user or an app permission to access one or more of the resources under your control.

You typically see this with integration applications.

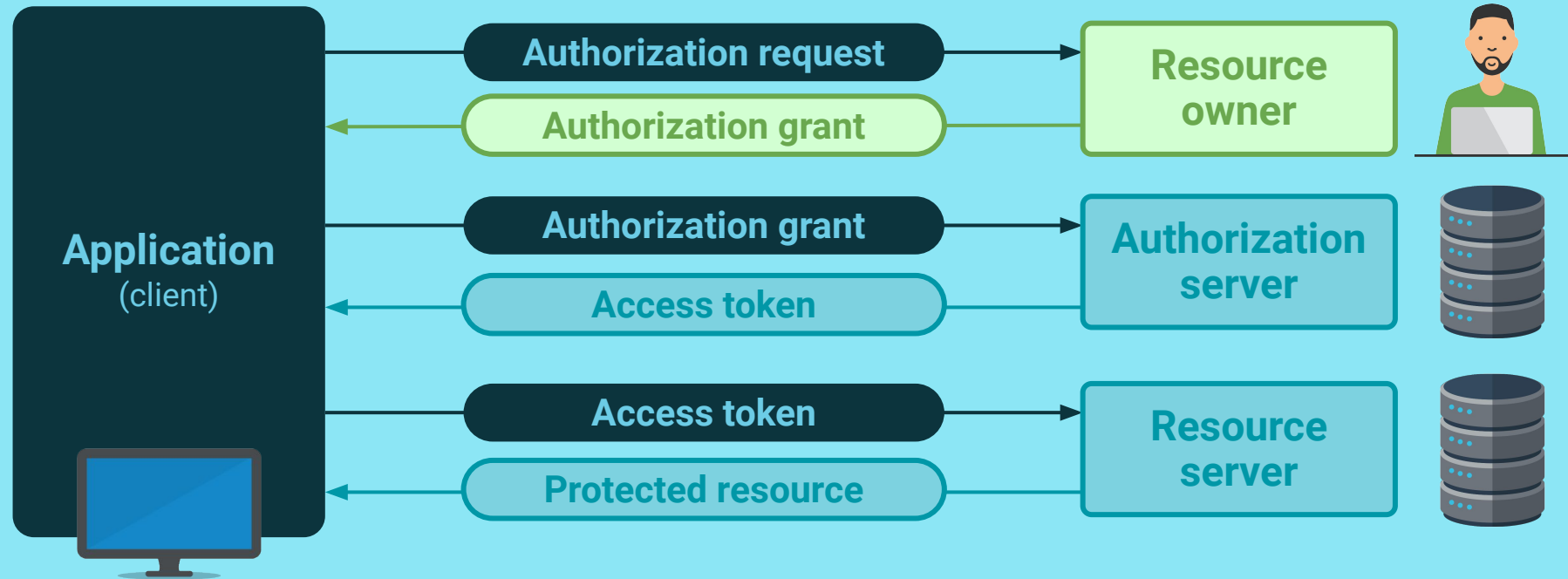
For example,

- You might have a task manager application that integrates with Google Calendar.
- This application may need permission to add events to your calendar.



Authorization Server

The authorization server is the part of an OAuth 2 system that issues the bearer tokens for users.

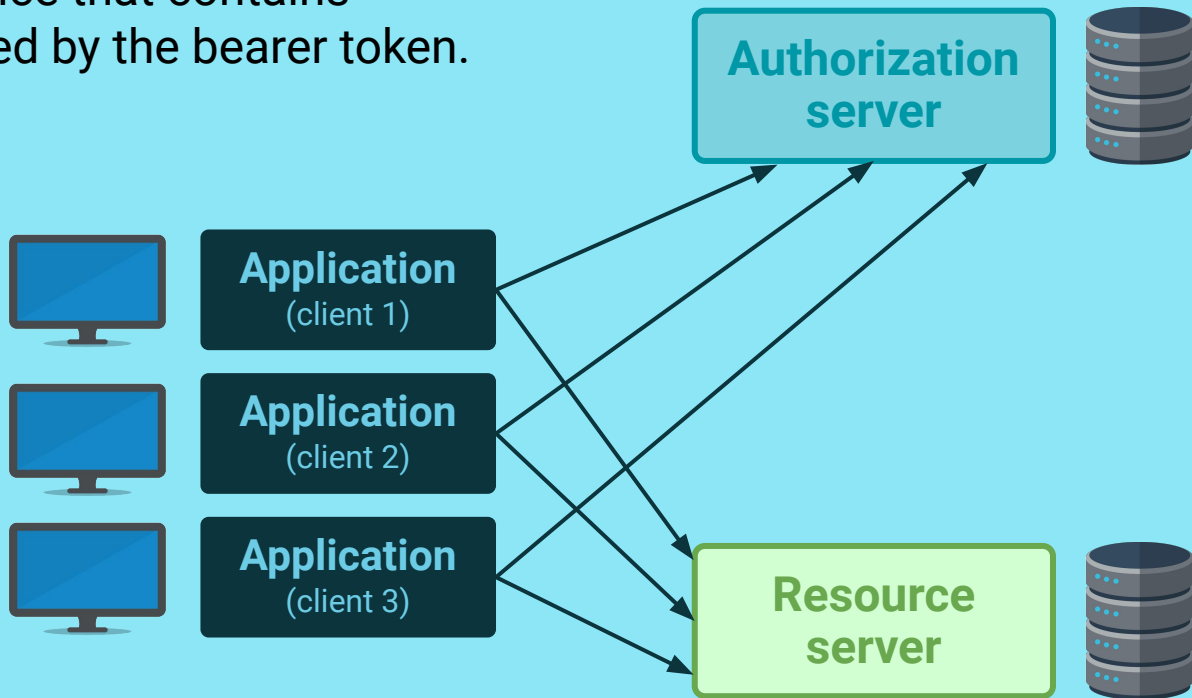


Resource Server

A resource server is a service that contains endpoints that are protected by the bearer token.

For example,

- It might be a separate microservice that can only be accessed with a valid OAuth 2 bearer token.
- Several resource servers can share one authorization server.



Bearer Token



The access token used by OAuth 2.



These tokens are arbitrary and have no meaning in and of themselves.



Bearer tokens are issued by the authorization server when the client requests a token and presents valid authentication data.



The client then uses the bearer token to access to a resource server.



Because these token have no inherent meaning, the resource server presents the token to the authorization server in exchange for user information.



The resource server grants access to the requested endpoint if the token is valid.



The resource server can then use the returned user information to help process the request if necessary.

Grant Types

OAuth 2 has different processes for granting tokens based on different use cases.

01

Authorization code grant:

The client exchanges an authorization code for a token.

02

Password grant:

- The client exchanges a username/password for a token.
- This should only be used for granting access to your service's own application.
- The system should never allow third-party applications for gaining access to passwords.

03

Client credential grant:

The application is requesting access as itself, not on behalf of a user.

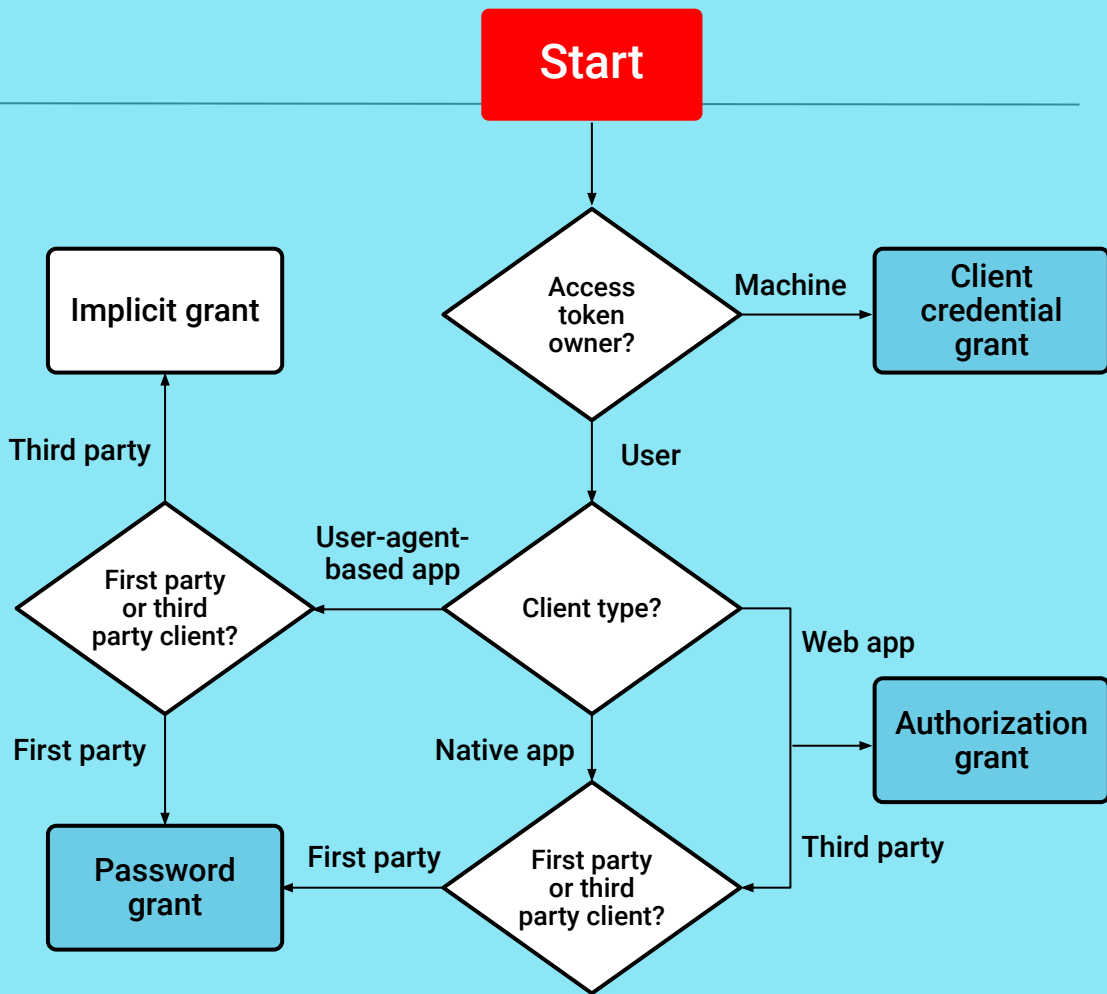
Grant Types

OAuth 2 has different processes for granting tokens based on different use cases.

01 Authorization grant

02 Password grant

03 Client credential grant



Clients

For example,

A client is a type of application that is allowed to access the system.



The ability to control how both clients and users that can access the system allows much finer grained control over access.

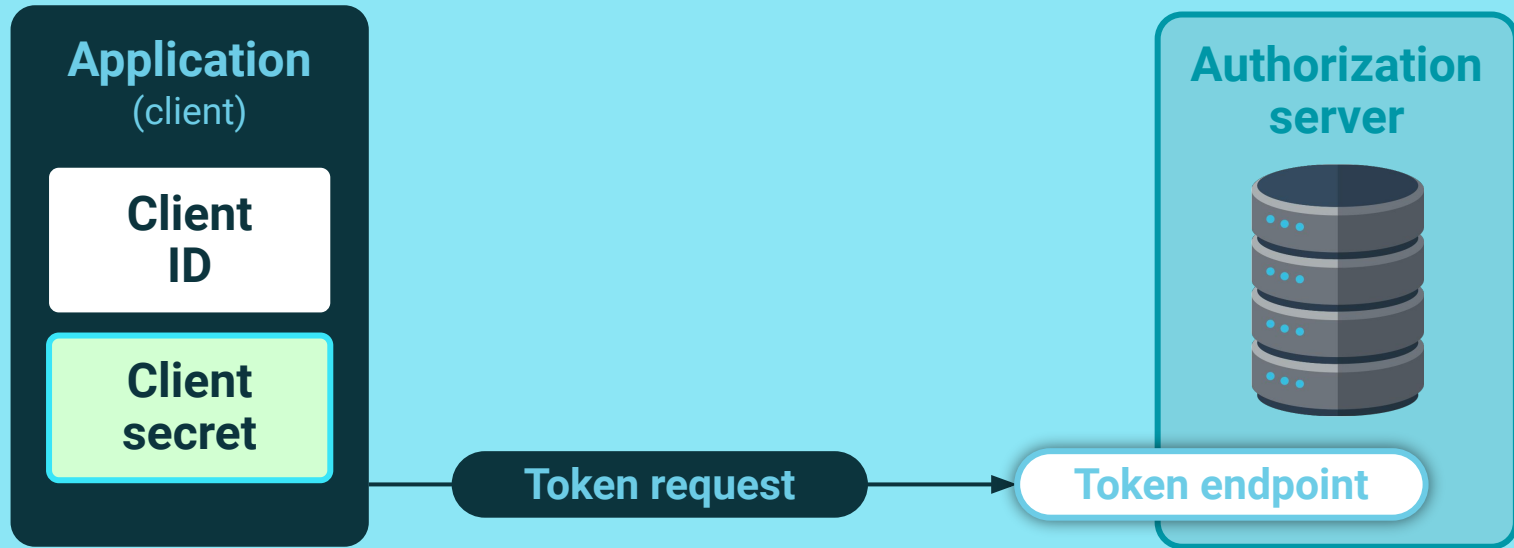
For example,

You could grant one level of access to Joe Smith using an HTML5 application and another level to Joe Smith using his smart phone.

Client Secrets

The client secret is like a password for the given client.

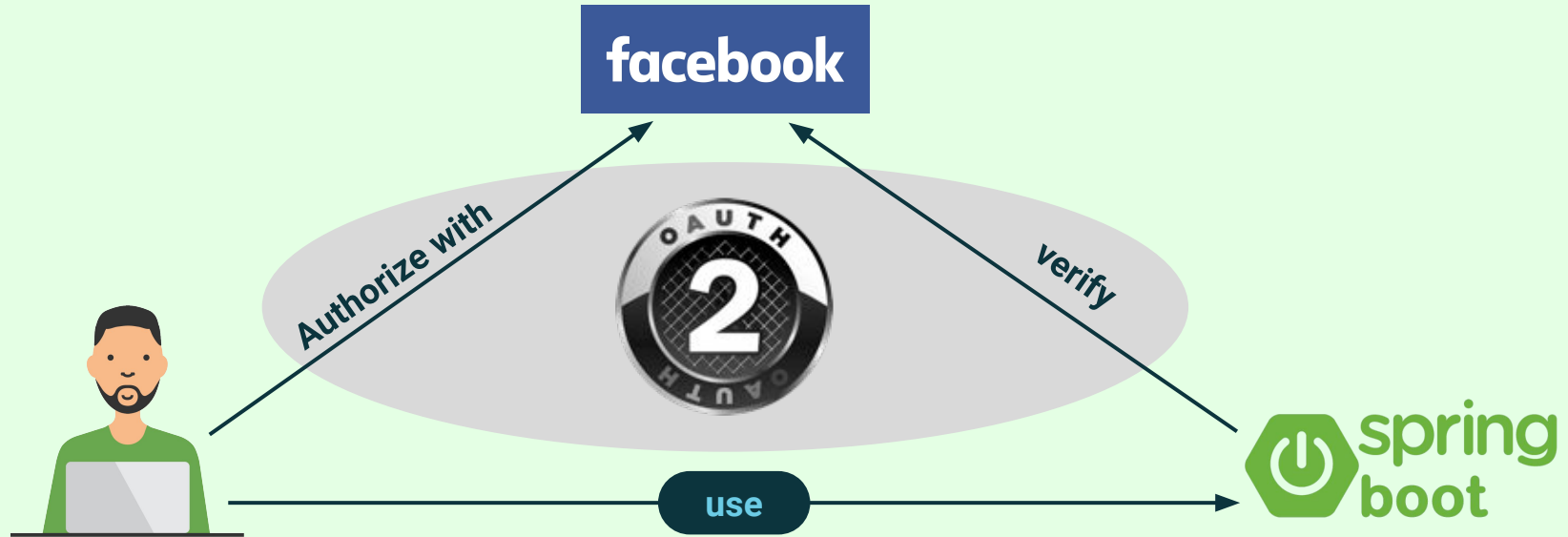
The correct client secret must be supplied along with the correct client specification in order to grant access.



Spring OAuth 2 Concepts

Authorization Server

This is a stand alone Spring Boot service that serves OAuth 2 bearer tokens to clients. This service also validates tokens presented by a resource server and sends back the user data associated with the given token.



Resource Server



This can be any Spring Boot web service or application.



Adding the `@EnableResourceServer` annotation turns a web service into a resource server.

Authentication Server

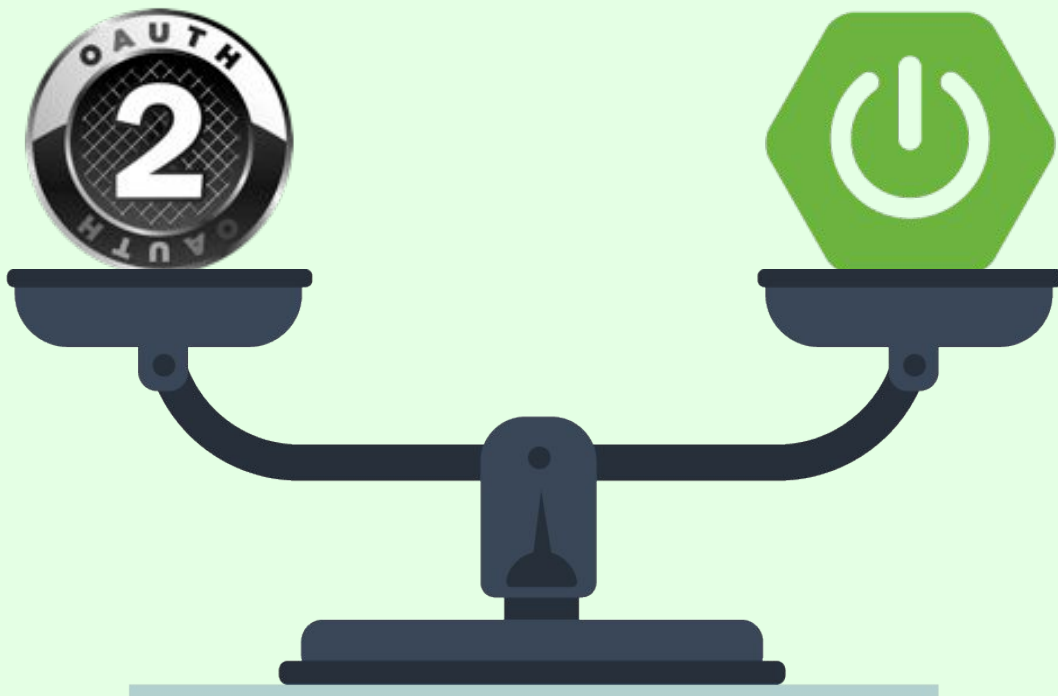
The Authentication Manager plays the same role in an OAuth 2 system as it does in a non-OAuth 2 Spring Security system.

The authentication manager is used by the authorization server to authenticate users for incoming requests.



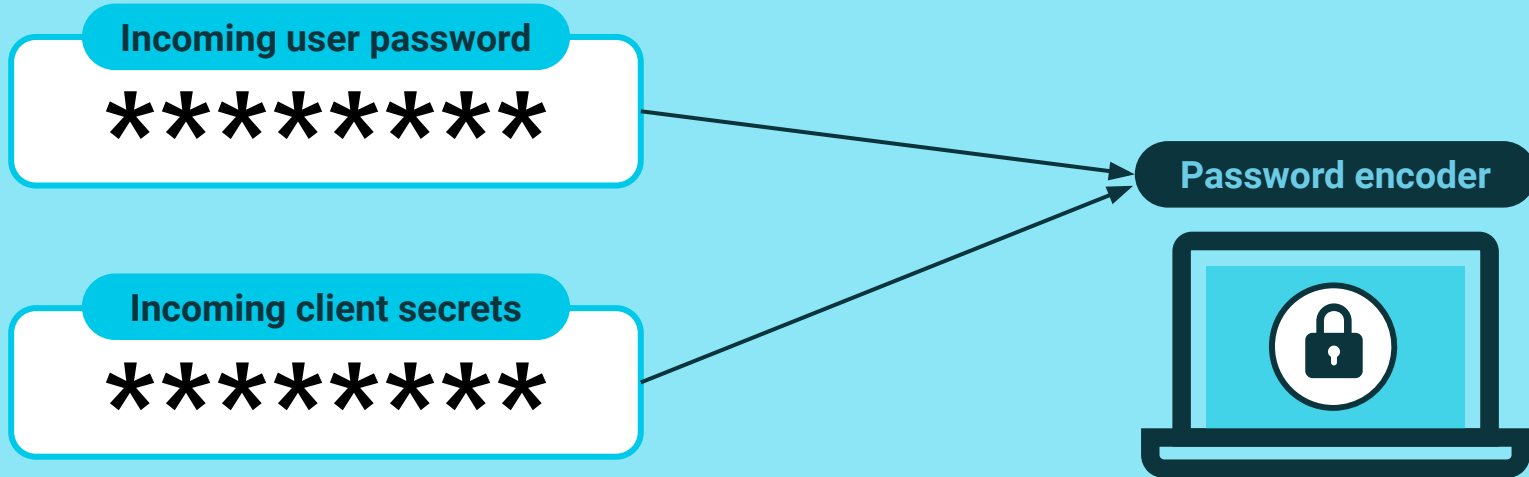
Authorities

Authorities play the same role in an OAuth 2 system as the do in a non-OAuth 2 Spring Security System.



Password Encoder

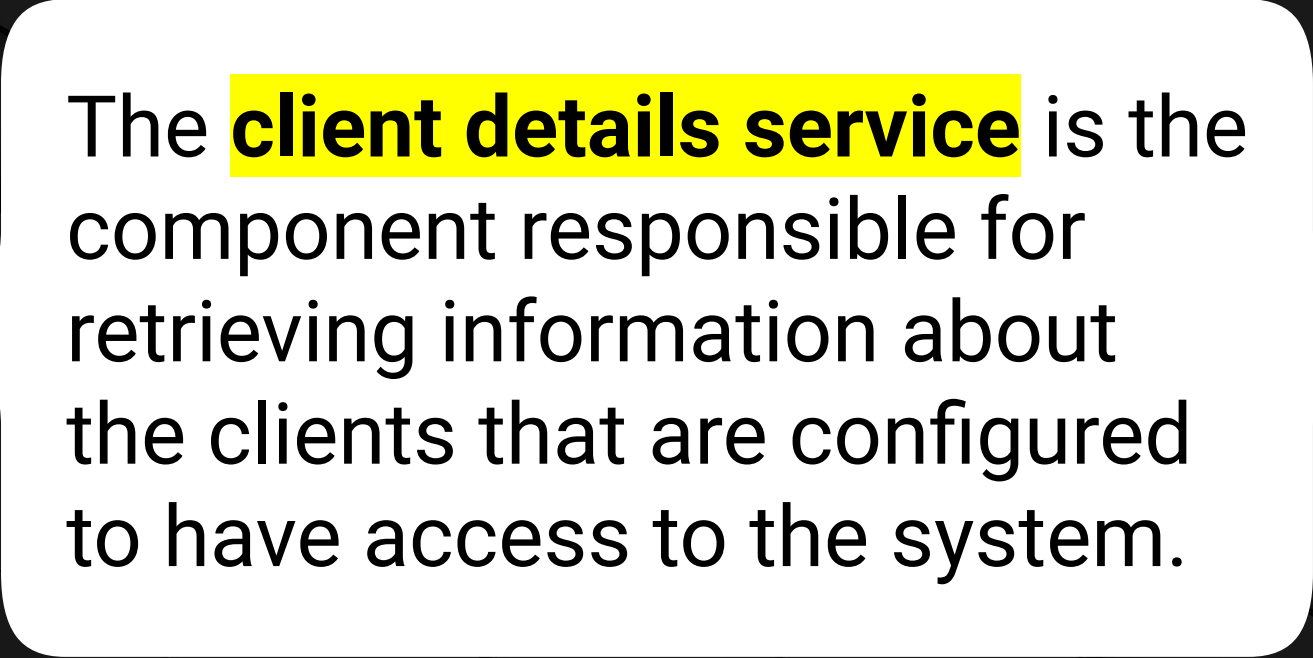
The password encoder is used to encode both incoming user password values and incoming client secrets before they are compared to the values stored in the system.



Client

The type of application that is allowed to access the system.
These are configured when the Client Details Service is configured.



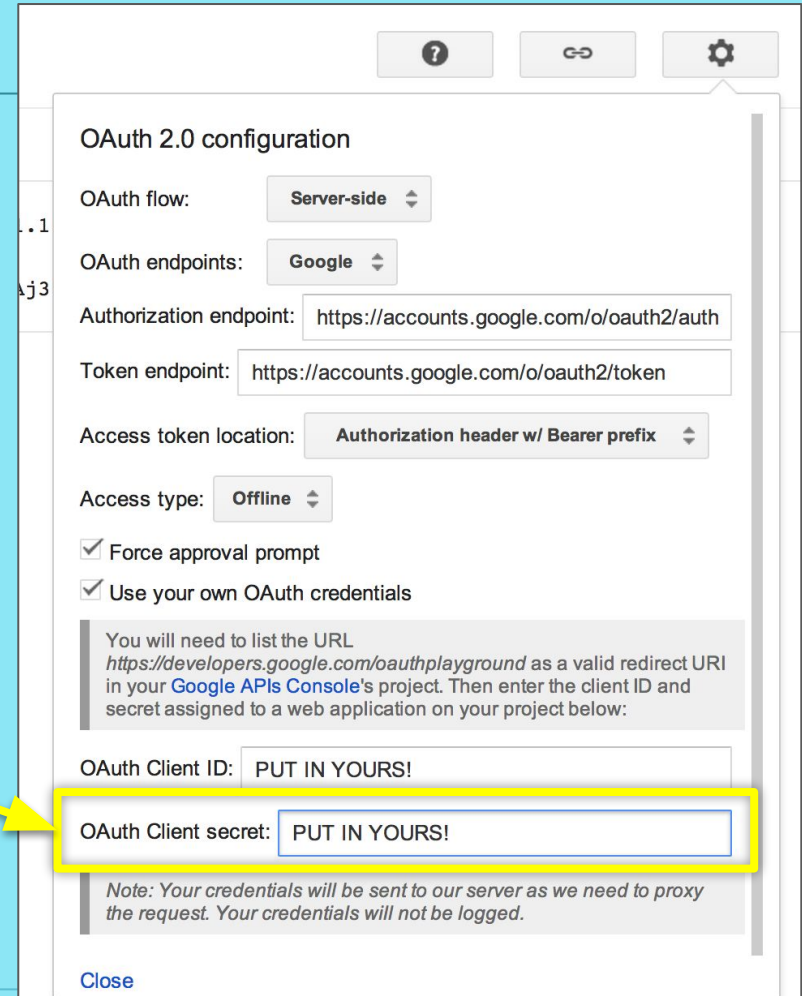


The **client details service** is the component responsible for retrieving information about the clients that are configured to have access to the system.

Client Secret

As mentioned above, this is like the password for the associated client.

The client secret is set when the Client Details Service is being created, configured, and populated.



OAuth 2.0 configuration

OAuth flow: **Server-side**

OAuth endpoints: **Google**

Authorization endpoint: `https://accounts.google.com/o/oauth2/auth`

Token endpoint: `https://accounts.google.com/o/oauth2/token`

Access token location: **Authorization header w/ Bearer prefix**

Access type: **Offline**

☒ Force approval prompt

☒ Use your own OAuth credentials

You will need to list the URL `https://developers.google.com/oauthplayground` as a valid redirect URI in your [Google APIs Console](#)'s project. Then enter the client ID and secret assigned to a web application on your project below:

OAuth Client ID: **PUT IN YOURS!**

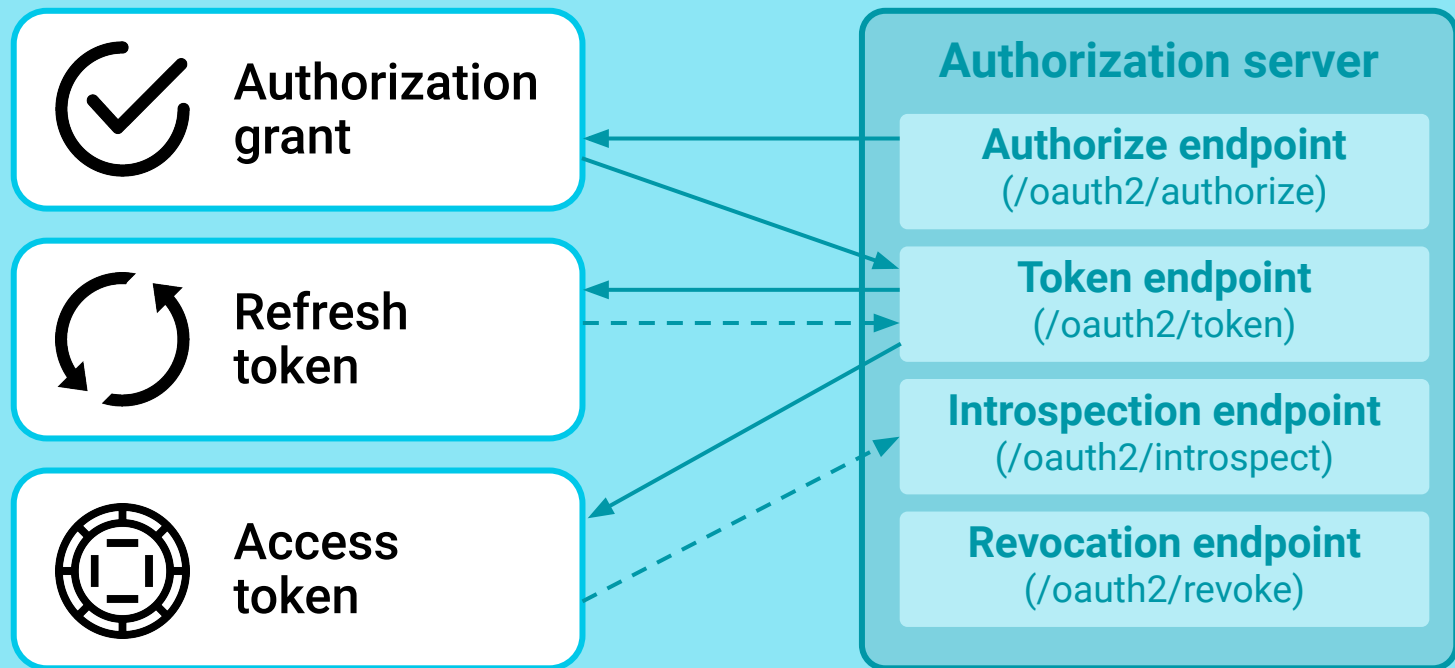
OAuth Client secret: **PUT IN YOURS!**

Note: Your credentials will be sent to our server as we need to proxy the request. Your credentials will not be logged.

Close

User Info Endpoint

This is an endpoint on the authorization server that allows resource servers to exchange valid tokens for the associated user data.





Time to Code

OAuth 2 Tutorial

Suggested Time:

1 – 2 hours

STOP
AND
THINK

STOP and THINK!

OAuth Concepts



What is federated authentication? What are some examples?



What is single sign-on? Why is it important to enterprises?



What is delegated authorization? What are some use cases?



What are bearer tokens? How are they used?



OAuth and Spring Security:

- What features of Spring Security can OAuth supply?
- How can OAuth and Spring Security work together?
- Are there situations where OAuth replaces Spring Security?



*The
End*