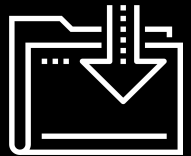


Common Authentication Schemes

Java Accelerator 7

Lesson 6.3



The background is a dark charcoal gray with a series of parallel diagonal lines running from the top-left to the bottom-right. Overlaid on this are several teal-colored geometric shapes: a large central triangle pointing right, a smaller triangle to its left, and a square to its right. Scattered around these shapes are various white line-art symbols, including a plus sign, a minus sign, a circle with a dot, a circle with a horizontal line, a circle with a vertical line, a circle with a diagonal line, a circle with a cross, a circle with a dot, a circle with a horizontal line, a circle with a vertical line, a circle with a diagonal line, a circle with a cross, a circle with a dot, a circle with a horizontal line, a circle with a vertical line, a circle with a diagonal line, and a circle with a cross.

WELCOME

Learning Outcomes

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

01

Use server-side OAuth with Spring Security.

02

Use server-side JWT with Spring Security.

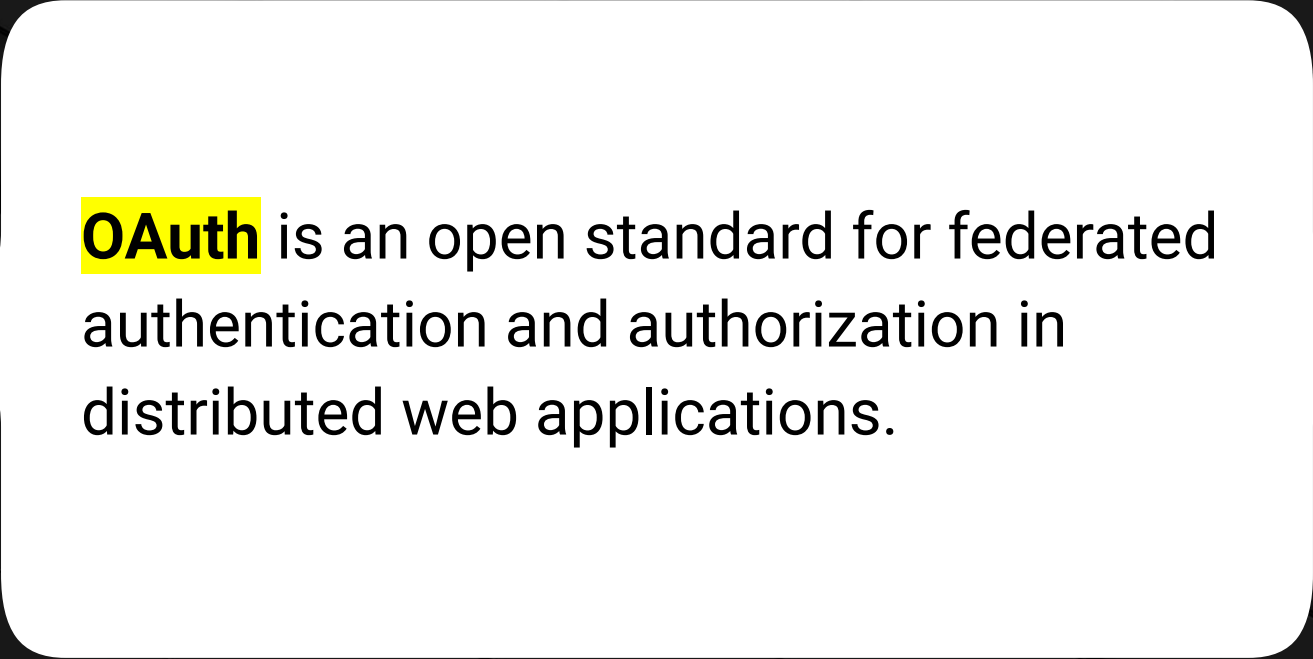


**How comfortable are you giving out
your login information to someone?**



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?

OAuth Basics



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

What is OAuth?



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



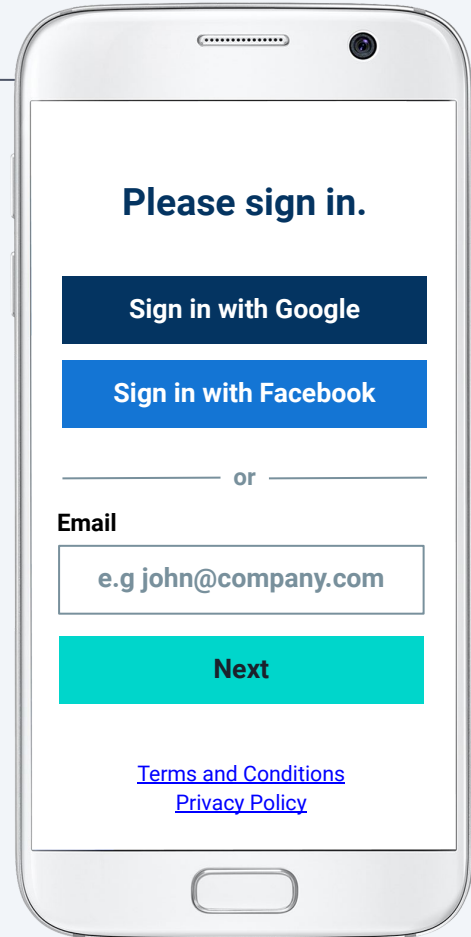
OAuth can also be used for single sign-on and distributed access for cloud-native microservice-based systems.

Federated Authentication

In **federated authentication**, one system delegates the responsibility of authentication to another system.

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

You've probably encountered this with "Log in with Facebook" or "Log in with your Google account" on various websites.



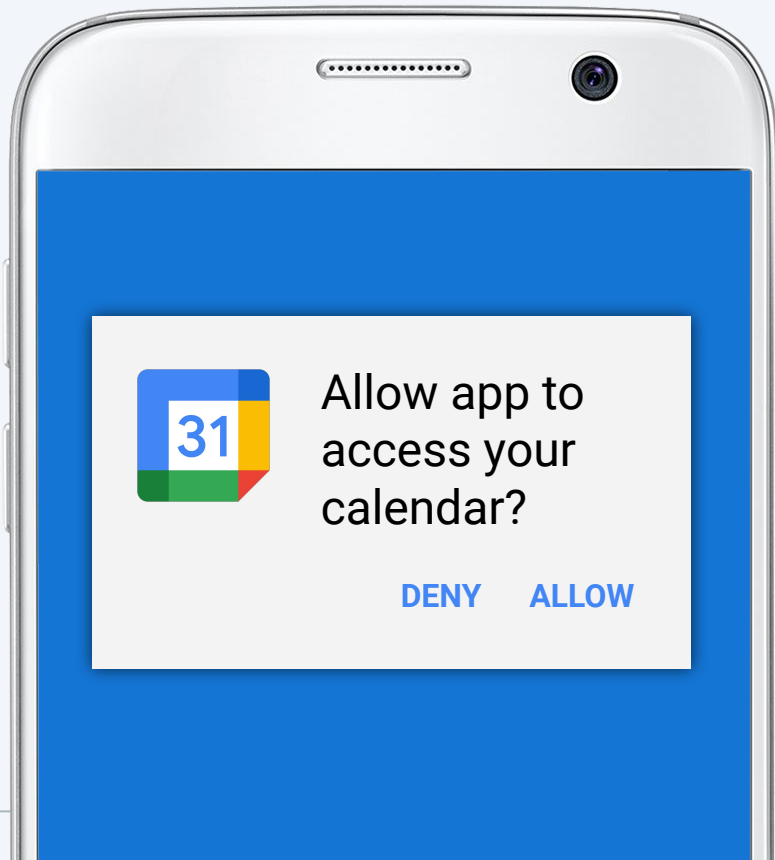
Delegated Authorization

Delegated authorization is giving another user or an app permission to access one or more of the resources under your control.

We typically encounter this with integration applications.

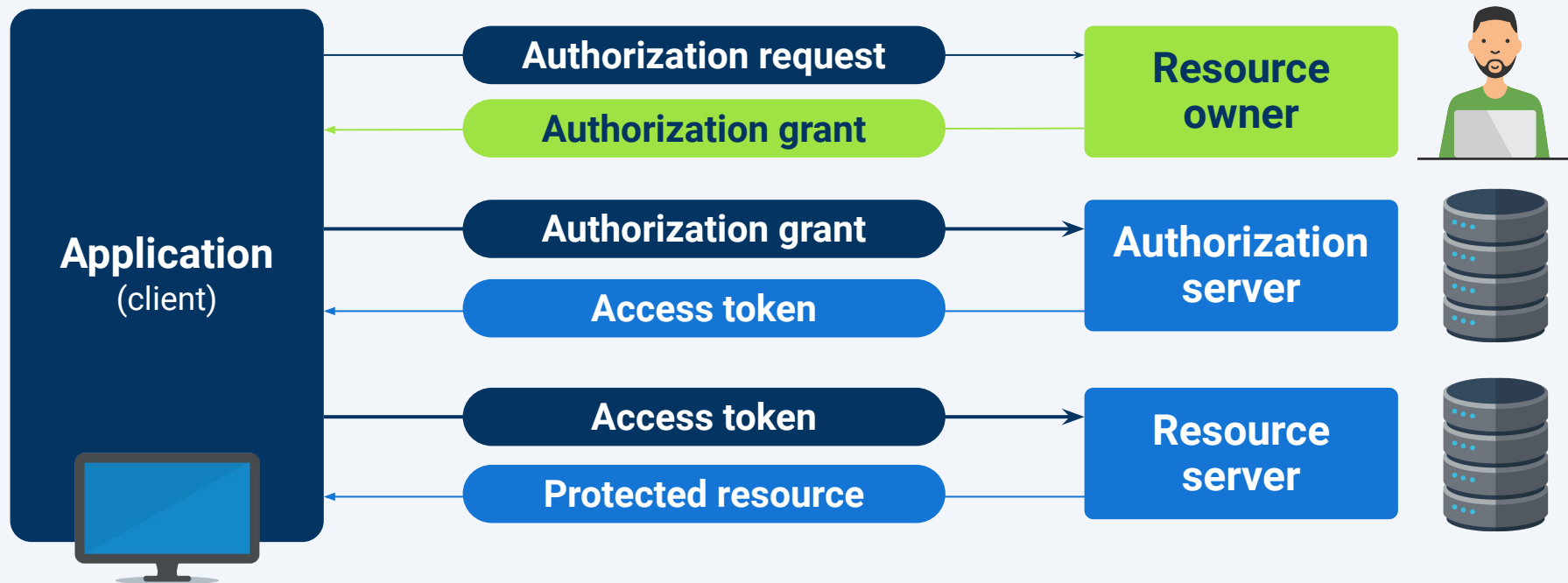
For example:

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



Authorization Server

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

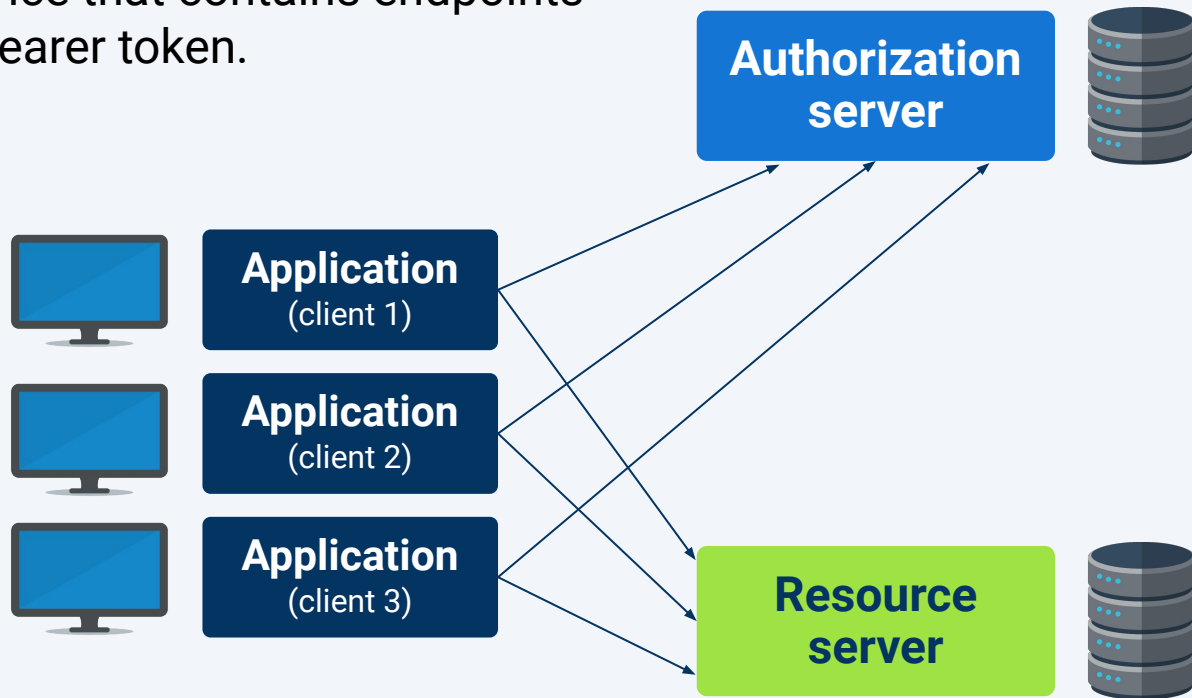


Resource Server

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



A **bearer token** is 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 a resource server.



Because these tokens 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.

Authorization Code Grant	<ul style="list-style-type: none">• The client exchanges an authorization code for a token.
Password Grant	<ul style="list-style-type: none">• The client exchanges a username/password for a token.• Should only be used for granting access to your service's own application.• The system should never give third-party applications access to passwords.
Client Credential Grant	<ul style="list-style-type: none">• The application is requesting access as itself, not on behalf of a user.

Clients

For example:

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



chrome



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

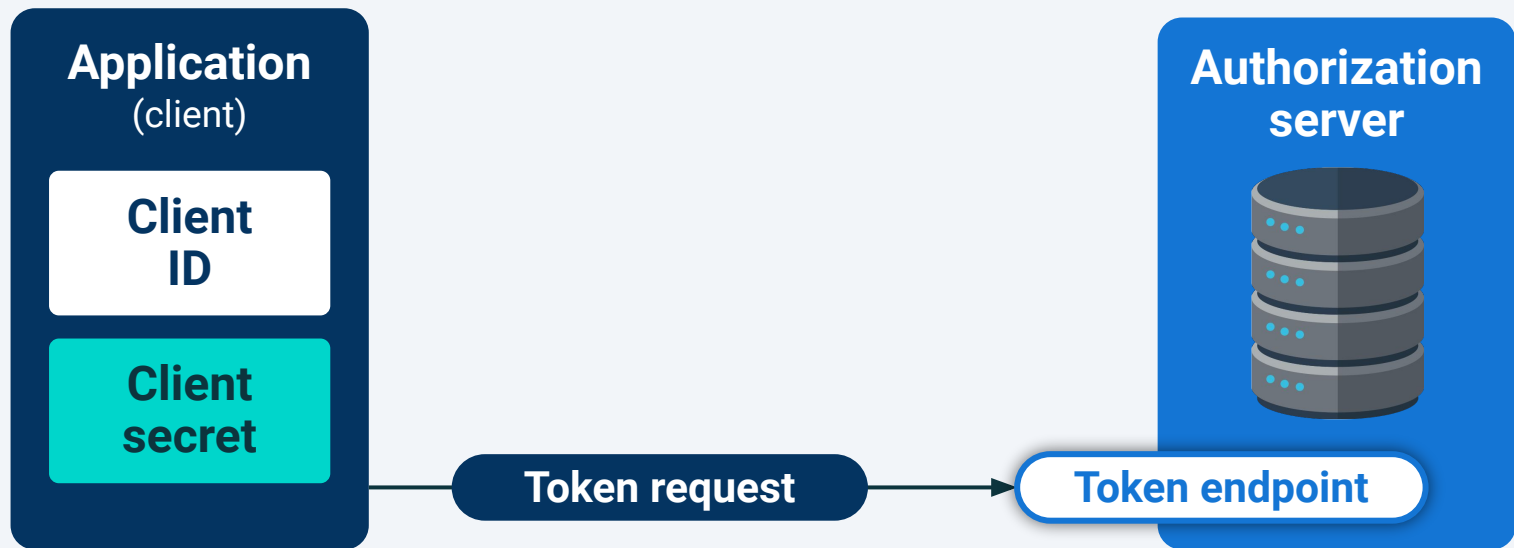
For example:

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

Client Secrets

Essentially, a **client secret** is a password for the given client.

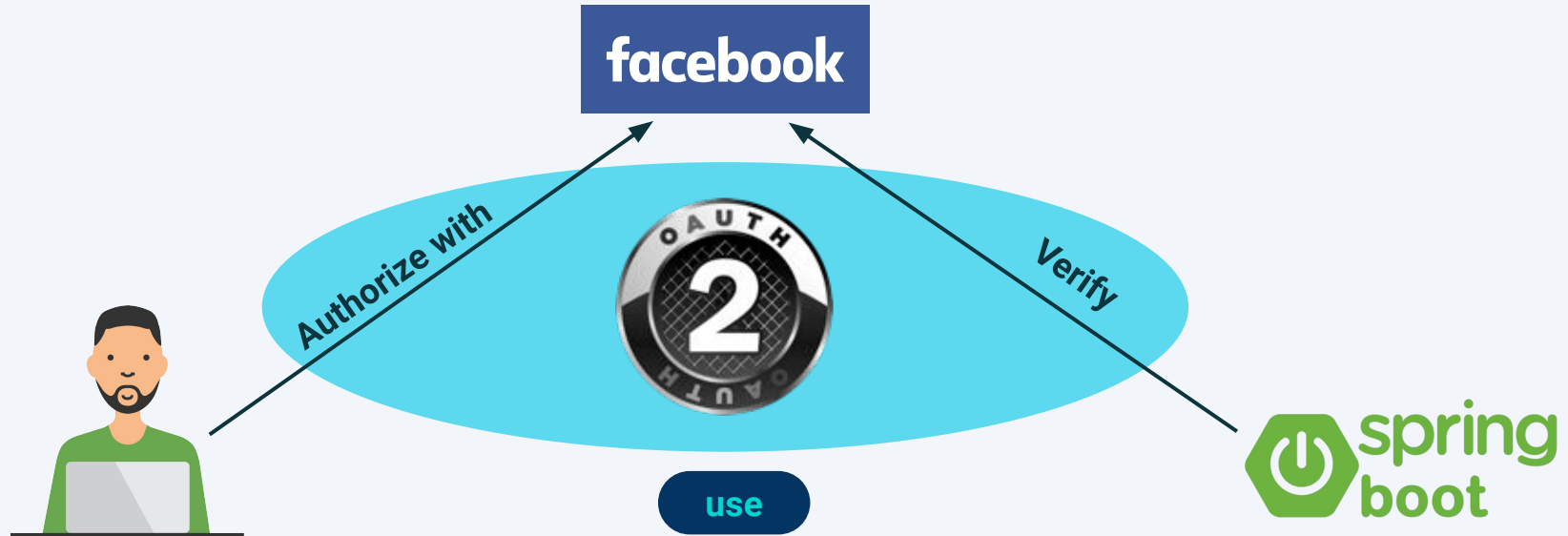
To grant access, the correct client secret must be supplied along with the correct client specification.



Spring OAuth Concepts

Authorization Server

A standalone 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



Can be any Spring Boot web service or application.



Authorization servers can also be resource servers.



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

Authentication Manager

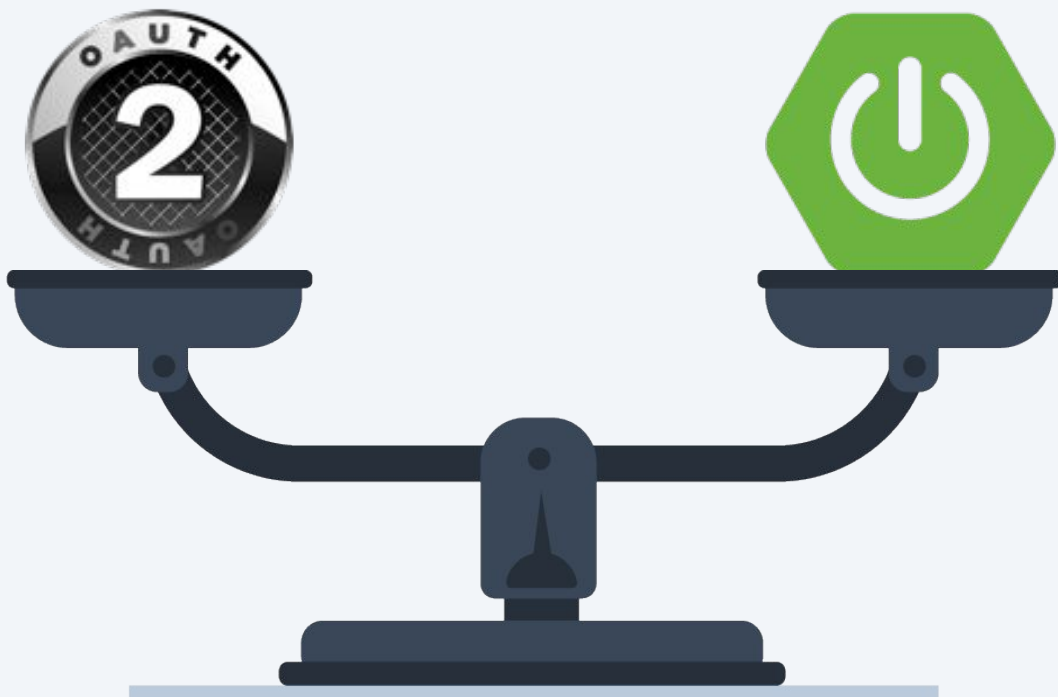
The authentication manager plays the same role in an OAuth 2 system as it does in a non-OAuth 2 Spring Security system.

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



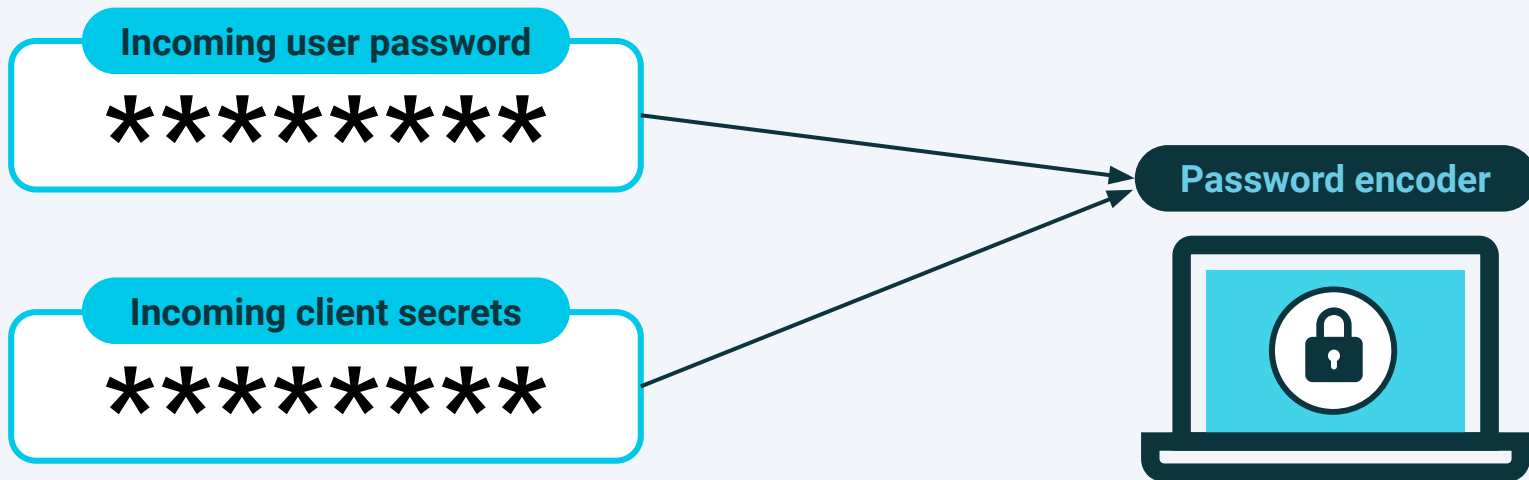
Authorities

Authorities play the same role in an OAuth 2 system as they 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're compared to the values stored in the system.

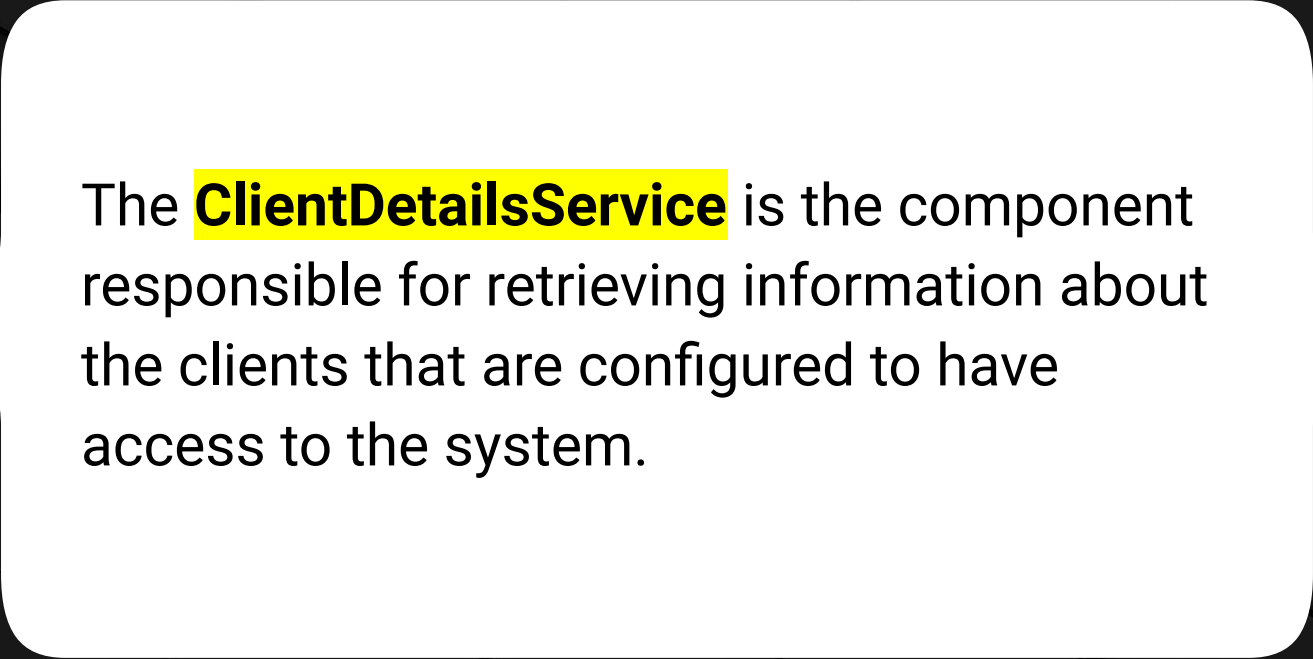


Client

Remember, the **client** is the type of application that is allowed to access the system.

These are configured when the `ClientDetailsService` is configured.



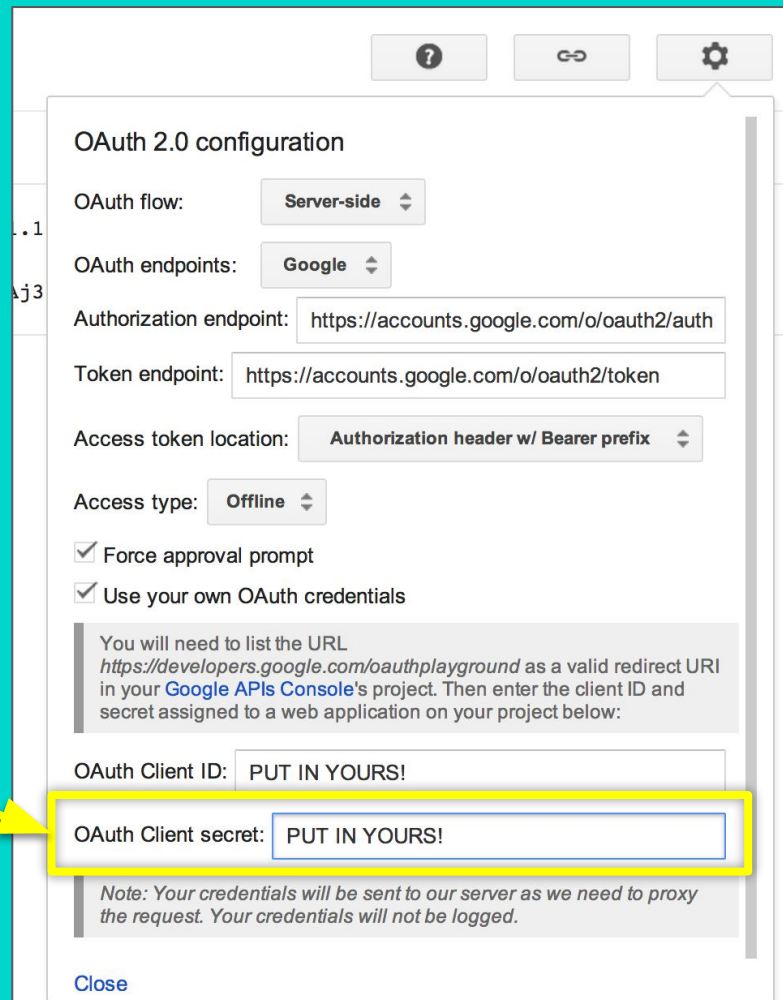


The **ClientDetailsService** is the component responsible for retrieving information about the clients that are configured to have access to the system.

Client Secret

The password for the associated client.

The client secret is set when the ClientDetailsService 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!**

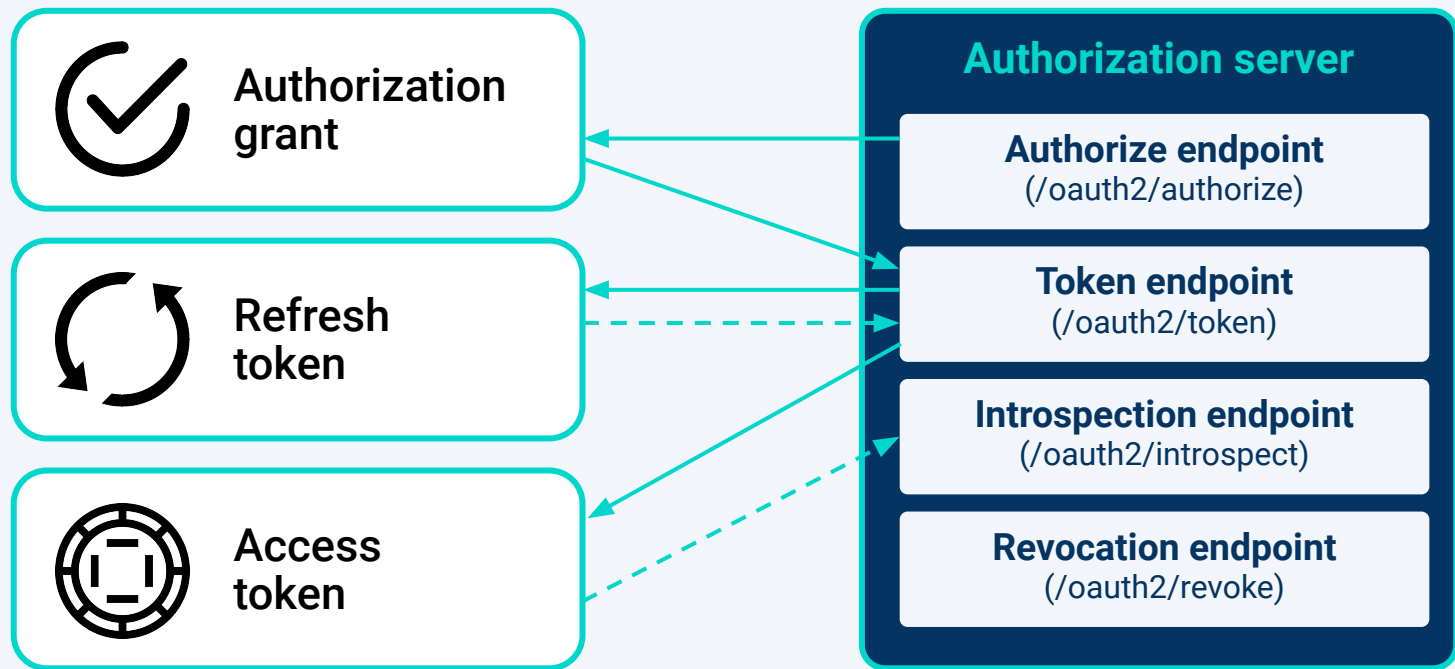
OAuth Client secret: **PUT IN YOURS!**

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.



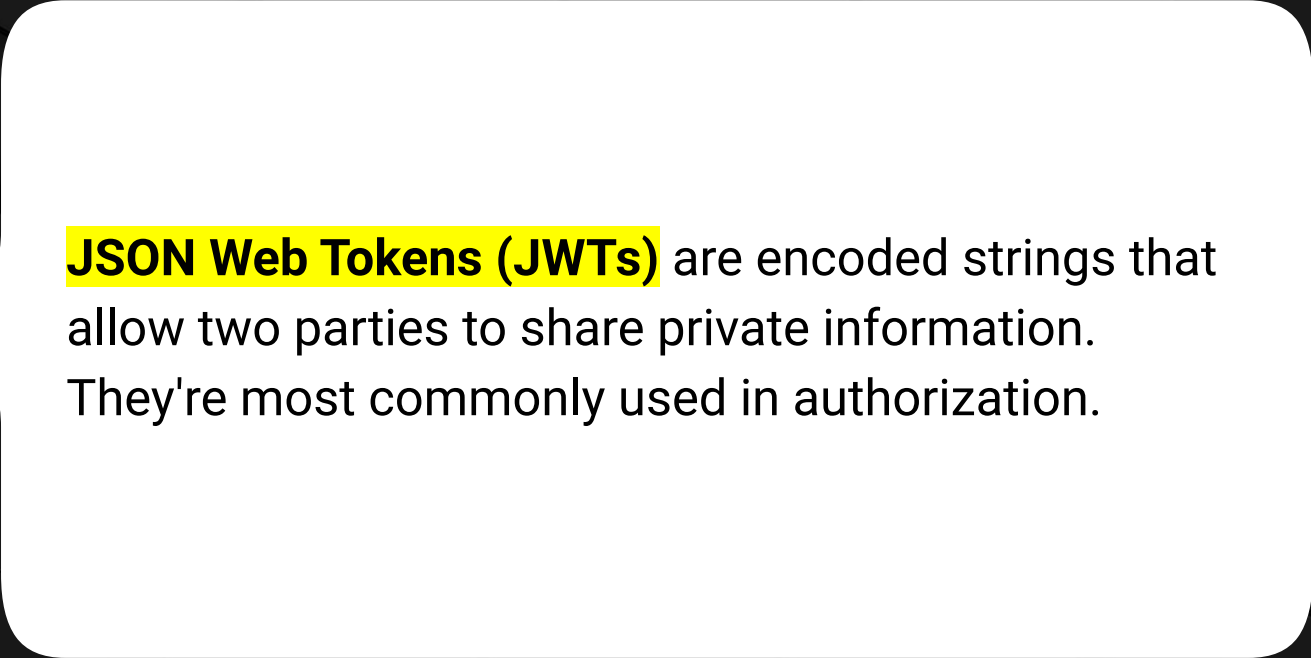


Time to <code>

JSON Web Tokens (JWT)



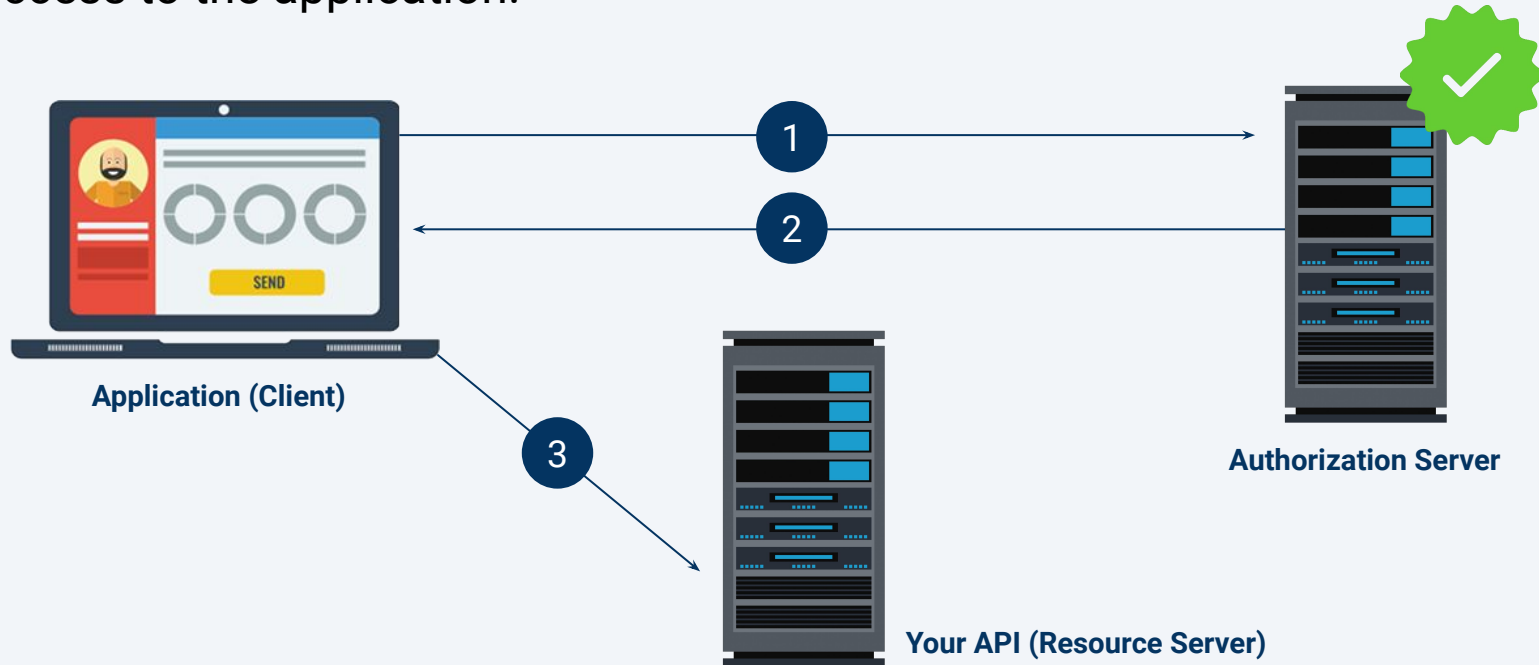
**Basic authentication isn't sufficient
when building modern web applications.
We need something more powerful for
front-end frameworks.**



JSON Web Tokens (JWTs) are encoded strings that allow two parties to share private information. They're most commonly used in authorization.

JSON Web Tokens (JWTs)

JWTs give us a more efficient way to implement security in a modern web application. A JWT is a type of bearer token. The bearer of the token has some access to the application.



JWT Structure

A JWT has three parts:

01

Header

02

Payload

03

Signature



The token is Base64 encoded and then either hashed or encrypted.

JWT Process



The client submits username/password credentials.



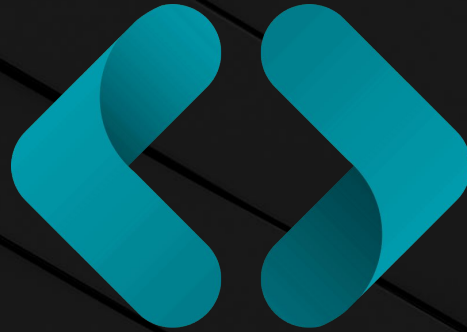
The server will authenticate a user by using username/password credentials.



The server will issue a JWT token if the credentials are valid.



The client must include the JWT token in with every subsequent request to the service.



Time to <code>