### oauth2 and okta auth are same

In summary, while OAuth 2.0 is a protocol for authorization, Okta is a service that implements this protocol among other identity management functionalities. You can use Okta as your authorization server when implementing OAuth 2.0 in your applications, leveraging its features to simplify the authentication process.

### different authorization server of OAuth 2

OAuth 2.0 authorization servers help applications securely authenticate users and access protected resources by issuing tokens that can be used for access control. Here are some widely used OAuth 2.0 authorization servers:

**1. Okta**

* **Overview**: Okta is an identity management service with robust OAuth 2.0 support, providing single sign-on (SSO), adaptive multi-factor authentication (MFA), and lifecycle management.
* **Features**: Supports OpenID Connect (OIDC), flexible role-based access control, and detailed analytics on login activity.
* **Use Cases**: Widely used by organizations needing enterprise-level identity management with easy integration options for both web and mobile applications.
* **Documentation**: Okta Documentation

**2. Auth0**

* **Overview**: Auth0 is a popular identity platform that supports OAuth 2.0 and OpenID Connect, with an emphasis on ease of integration for developers.
* **Features**: Customizable login pages, social logins, multi-factor authentication, and APIs for machine-to-machine authorization.
* **Use Cases**: Ideal for startups and enterprises alike due to its flexibility and developer-friendly SDKs.
* **Documentation**: Auth0 Documentation

**3. AWS Cognito**

* **Overview**: Amazon Cognito provides OAuth 2.0 support as part of its authentication services, especially useful for applications hosted on AWS.
* **Features**: Allows federated identity with social providers, supports custom authentication flows, and integrates easily with AWS services.
* **Use Cases**: Frequently used by applications built on AWS, providing a low-cost and scalable option with built-in identity management.
* **Documentation**: [AWS Cognito Documentation](https://docs.aws.amazon.com/cognito/)

**4. Google Identity Platform**

* **Overview**: Google Identity provides OAuth 2.0 support, primarily through Google Sign-In, and enables access to Google APIs with delegated permissions.
* **Features**: Supports secure login for Google accounts, MFA, and integration with Google APIs.
* **Use Cases**: Often chosen for applications requiring access to Google services or that need to integrate with Google Workspace.
* **Documentation**: Google Identity Platform Documentation

**5. Azure Active Directory (Azure AD)**

* **Overview**: Microsoft’s Azure AD is a cloud-based identity platform with OAuth 2.0 support, often used in enterprise settings with Microsoft 365 and Azure integrations.
* **Features**: Offers enterprise-grade security, conditional access policies, and seamless SSO for Microsoft products.
* **Use Cases**: Suitable for organizations heavily using Microsoft services and wanting to manage access to multiple Microsoft and third-party applications.
* **Documentation**: [Azure AD Documentation](https://docs.microsoft.com/en-us/azure/active-directory/)

**6. Keycloak**

* **Overview**: Keycloak is an open-source identity and access management server that supports OAuth 2.0, OpenID Connect, and SAML.
* **Features**: Offers SSO, social login, LDAP integration, and highly customizable authentication flows.
* **Use Cases**: Popular with organizations needing an on-premises or self-hosted solution with flexibility in customization.
* **Documentation**: Keycloak Documentation

**7. Ping Identity**

* **Overview**: Ping Identity provides an enterprise-grade identity solution with OAuth 2.0 and OpenID Connect support, focusing on secure and scalable access management.
* **Features**: Supports MFA, risk-based access policies, and advanced access control.
* **Use Cases**: Commonly used by large enterprises for secure access across cloud and on-premise environments.
* **Documentation**: Ping Identity Documentation

**Choosing the Right OAuth 2.0 Server**

The best authorization server depends on your application requirements, deployment preferences, and budget. For instance:

* **Small to Medium Enterprises** may prefer Auth0 or AWS Cognito for ease of integration and scalability.
* **Enterprises** often choose Okta, Ping Identity, or Azure AD due to their focus on robust security and compliance.
* **Self-Hosted Requirements** are well suited to Keycloak, which provides flexibility and control over data and customization.

For more information, refer to the official documentation of each provider listed above to see which one aligns best with your specific needs and integration requirements.

### Ergo UI – This will be based on user id and password

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Employee detail

A screenshot of a computer

Description automatically generated

How to get token for Ergo svc from UI

A screenshot of a computer

Description automatically generated

How ergo svc is secure

### spring.security.oauth2.resource-server.jwk-set-uri

The spring.security.oauth2.resource-server.jwk-set-uri property in Spring Security is used to specify the URI where the JSON Web Key Set (JWKS) can be found. This is particularly useful for configuring a resource server that needs to validate JWTs (JSON Web Tokens) issued by an authorization server.

Here’s how you can configure it in your application.yml or application.properties file:

A screenshot of a computer

Description automatically generated

**Explanation:**

* **jwk-set-uri**: This should point to the location of the JWKS, which is a JSON document containing public keys that can be used to verify the signature of the JWTs. The JWKS is typically provided by the authorization server and may be found at a well-known URL.

**Usage:**

1. **Set Up Resource Server**: When you configure a Spring application as a resource server, this property allows it to automatically fetch the public keys needed to validate incoming JWTs.
2. **Security Configuration**: Ensure your security configuration class is set up to enable resource server capabilities.

**Example Security Configuration**

A screenshot of a computer

Description automatically generated

**Important Notes:**

* Ensure that the URL specified in jwk-set-uri is accessible by your application.
* The JWKS should include the necessary public keys to verify the JWTs issued by your authorization server.
* Make sure to handle exceptions and implement proper error handling for cases when the JWKS is not reachable.

This configuration allows your application to authenticate requests securely using JWTs signed by the specified authorization server.

A screenshot of a computer screen

Description automatically generated