

Secure Order History Backend



View Order History

- The Order History functionality is working
- However, the REST API for /api/orders is unsecured

The screenshot shows the luv2shop website interface. At the top, there is a navigation bar with the logo 'luv2shop', a search bar, and user information 'Welcome back Demo Darby!'. The 'Orders' button in the navigation bar is highlighted with a red dashed circle and a red arrow pointing to it from the bottom right. Below the navigation bar, the main content area is titled 'Your Orders' and displays a table with two rows of order data.

Order Tracking Number	Total Price	Total Quantity	Date
51542a04-e16d-4e3e-9549-7e3394558f8d	\$36.98	2	Feb 13, 2021, 10:39:57 AM
0e8014f2-0cd9-4ce4-bb5a-e451154ee9f0	\$17.99	1	Feb 13, 2021, 11:59:58 AM

Secure Order History Backend

- Let's secure the backend
- `/api/orders` should only available to logged in users

HTTP Method	Endpoint
GET	<code>/api/orders/search/findByCustomerEmail?email=demo@luv2code.com</code>

Development Process - Spring Boot

Step-By-Step

1. Add Okta Spring Boot Starter to Maven pom.xml
2. Create an App at the Okta Developer website
3. In Spring Boot app, set application properties
4. Protect endpoints in Spring Security configuration class

Step 1: Add Okta Spring Boot Starter to Maven pom.xml

- Okta provides a Spring Boot Starter for OAuth 2 / OpenID Connect
- Simplifies integration and configuration of Spring Security and Okta

File: pom.xml

```
...
<dependency>
    <groupId>com.okta.spring</groupId>
    <artifactId>okta-spring-boot-starter</artifactId>
    <version>2.0.1</version>
</dependency>
...
```

<https://github.com/okta/okta-spring-boot>

Step 2: Create an App in Okta Developer website

The image shows two screenshots of the Okta Developer website. The left screenshot is a modal titled "Create a New Application Integration". It has a "Platform" dropdown set to "Web", a "Sign on method" section with three options: "Secure Web Authentication (SWA)", "SAML 2.0", and "OpenID Connect" (which is selected), and "Create" and "Cancel" buttons. The right screenshot is a larger window titled "Create OpenID Connect App Integration". It has a "General Settings" section with an "Application name" field containing "Order History Backend", an "Application logo (Optional)" field with a "Browse files..." button, and a "Requirements" section with instructions for best results. Below this is a "Configure OpenID Connect" section with a "Login redirect URIs" field containing "http://localhost:8080/login/oauth2/code/okta" and a "+ Add URI" button. A red arrow points from the "Create" button in the first window to the "General Settings" section in the second window.

Create a New Application Integration

Platform: Web

Sign on method:

- Secure Web Authentication (SWA)
- SAML 2.0
- OpenID Connect

Create Cancel

General Settings

Application name: Order History Backend

Application logo (Optional) Browse files...

Requirements

- Must be PNG, JPG or GIF
- Less than 1MB

For Best Results, use a PNG image with

- Minimum 420px by 120px to prevent upscaling
- Landscape orientation
- Transparent background

Configure OpenID Connect

Login redirect URIs: http://localhost:8080/login/oauth2/code/okta

+ Add URI

After Okta authenticates a user's sign-in request, Okta redirects the user to one of these URIs

Step 3: In Spring Boot app, set application properties

File: application.properties

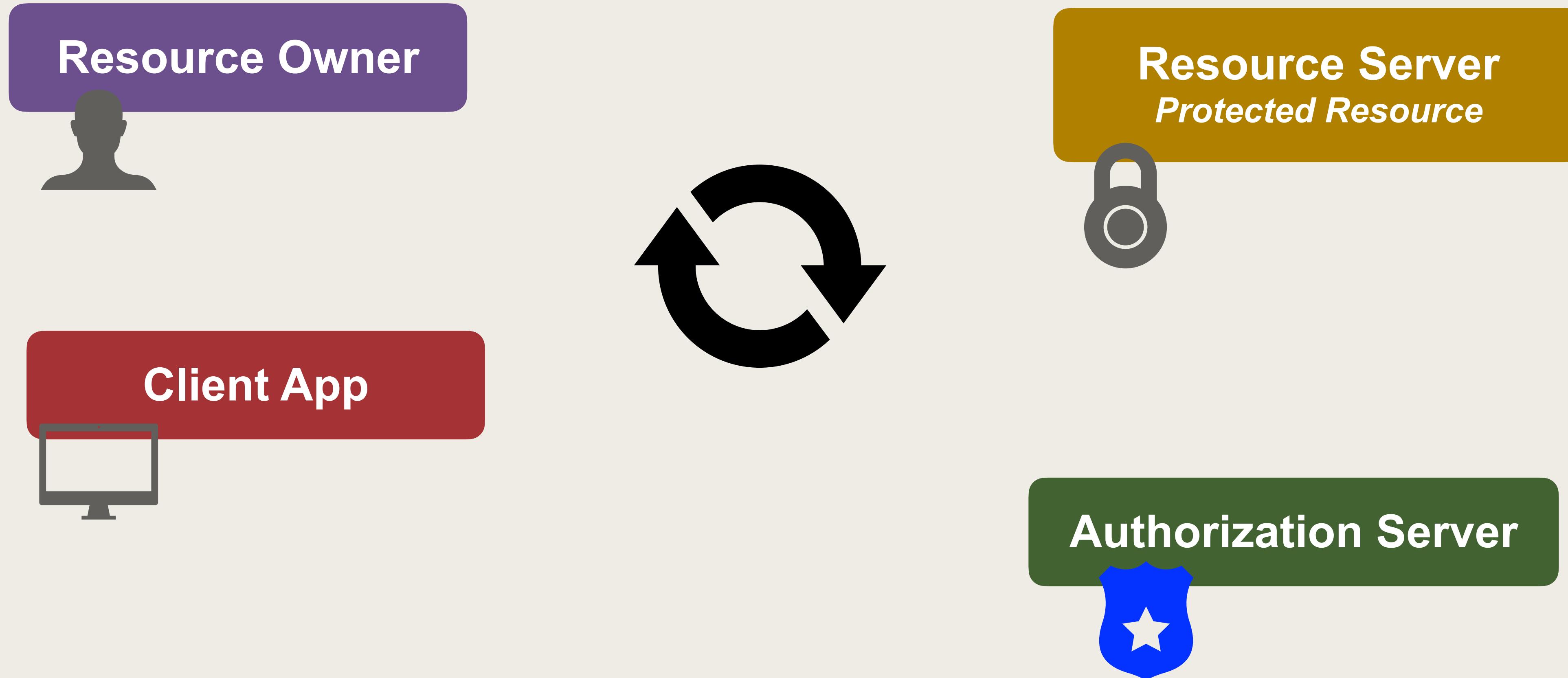
```
...  
okta.oauth2.client-id={yourClientId}  
okta.oauth2.client-secret={yourClientSecret}  
okta.oauth2.issuer=https:// {yourOktaDomain}/oauth2/default
```

Spring Boot application will use these properties to verify / validate JWT access tokens

The screenshot shows the 'Client Credentials' section of the Okta client configuration. It includes fields for 'Client ID' (set to 'Ooafdq3bw5B05TA1f5d6') and 'Client secret' (represented by a series of dots). Below this is a 'General Settings' section with the 'Okta domain' field set to 'dev-5389590.okta.com'. There are 'Edit' buttons next to each field.

Client Credentials	
Client ID	Ooafdq3bw5B05TA1f5d6
Client secret
General Settings	
Okta domain	dev-5389590.okta.com

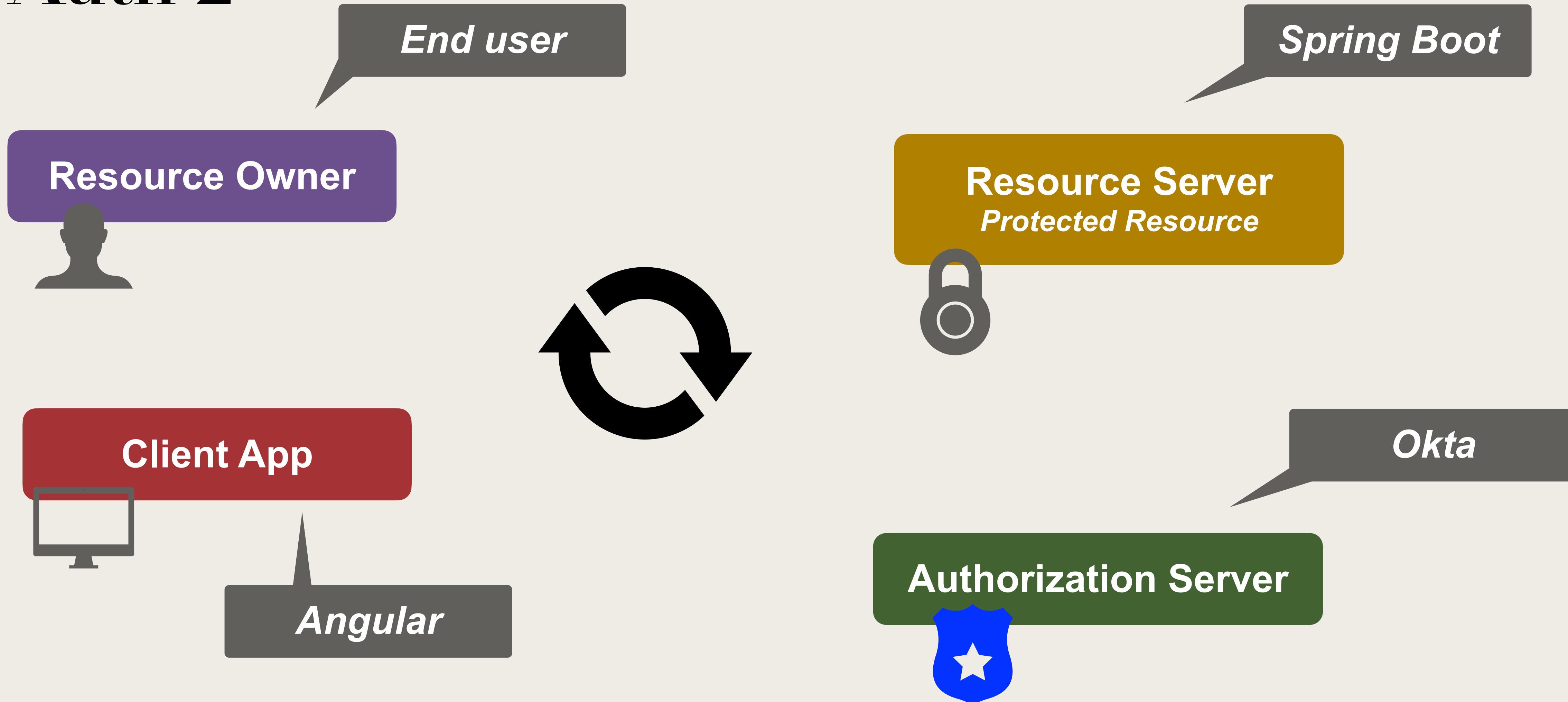
OAuth 2



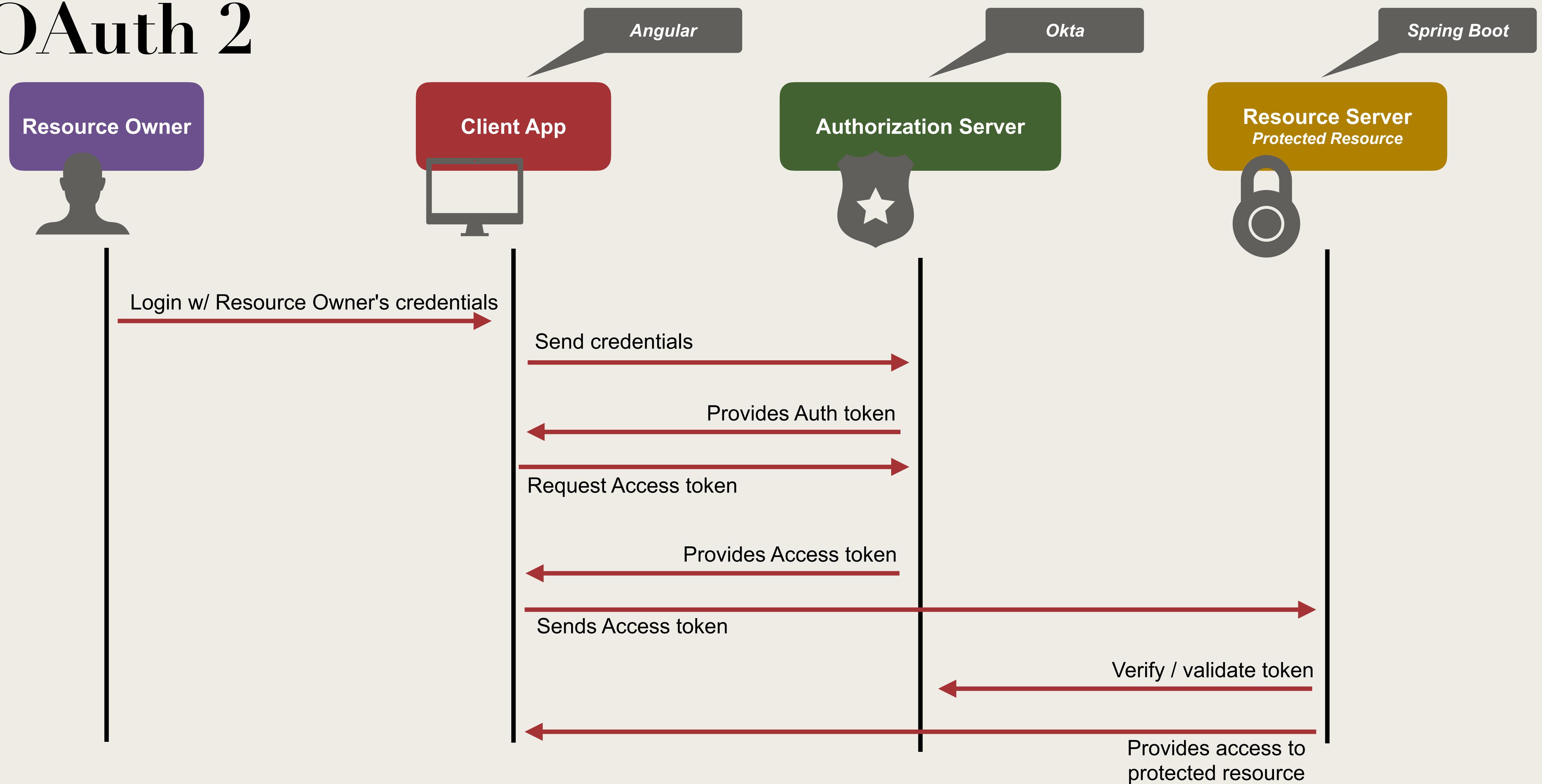
Resource Server

- The "Resource Server" is the app that is hosting our protected resources
 - In our case, it is our Spring Boot app
- The "Resource Server" manages security using access tokens (JWT)
- The access tokens are validated with the "Authorization Server" (Okta)

OAuth 2



OAuth 2



Client sending access token

- Client sends the access token as an HTTP request header

Request header

Authorization: Bearer <token>

Request body

....
....
....

Step 4: Protect endpoints in Spring Security configuration class

File: SecurityConfiguration.java

```
package com.luv2code.ecommerce.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // protect endpoint /api/orders
        http.authorizeRequests()
            .antMatchers("/api/orders/**")
            .authenticated()
            .and()
            .oauth2ResourceServer()
            .jwt();

        // add CORS filter
        http.cors();
    }
}
```

Protect the endpoint ... only accessible to authenticated users

Configures OAuth2 Resource Server support

Enables JWT-encoded bearer token support