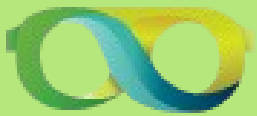


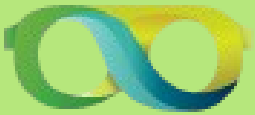
Securing Microservices using OAuth2

Presented By

Kalyan Mondal

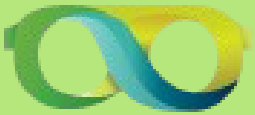
Karan Bhargava

Agenda



- What is Microservice?
- What is OAuth?
- Why use OAuth?
- Who is Using?
- OAuth Timeline
- OAuth1 vs OAuth2
- OAuth2 Roles
- OAuth2 Basic Flow
- OAuth2 Grant Types
- OAuth2 Tokens
- Pros and Cons
- Spring Security OAuth
- References

What is Microservice

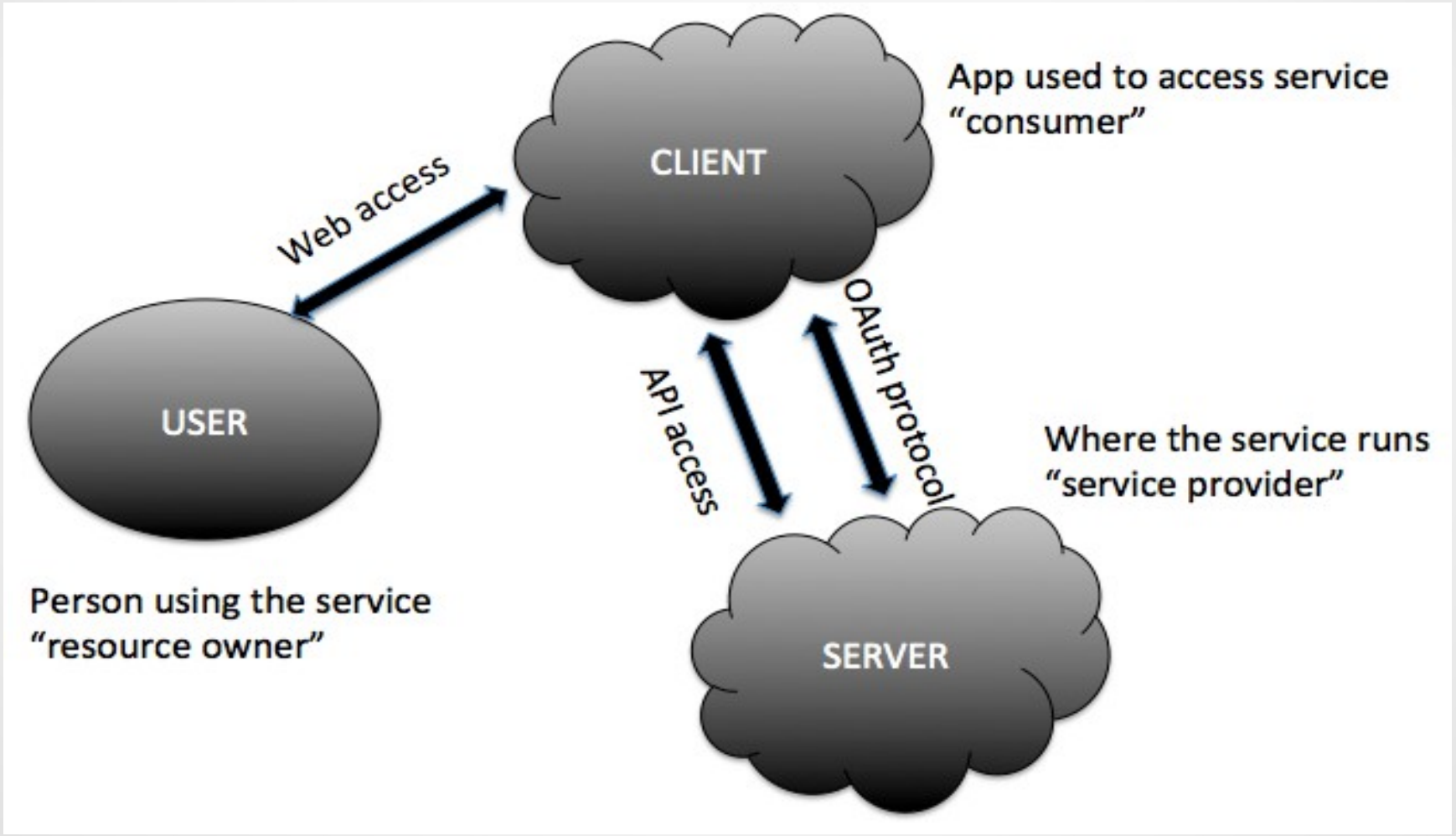


- HTTP Protocol
- Text based message content e.g. JSON
- Small, Compact and Quick Response message
- RESTful
- Statelessness

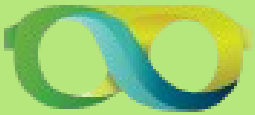
What is OAuth

- OAuth stands for “Open Authorization”
- Open standard protocol that provides Simple and Secure authorization
- Give access to users without any use of Credentials
- Use of Tokens
- Security and Authorization

OAuth


CLIENT
App used to access service
"consumer"
Web access
USER
Person using the service
"resource owner"
API access
OAuth protocol
SERVER
Where the service runs
"service provider"

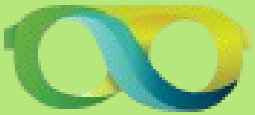
Why OAuth



- Securing RESTful API
- Enable Applications to access each others data without sharing the password
- Compatible to mobile devices and desktop applications
- Grant limited access in terms of scope and duration
- Increased Trust
- Easy to implement

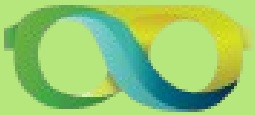
Who is using OAuth

OAuth Timeline



- OAuth1.0 – December, 2007
- OAuth1.0a – June, 2009
- OAuth2.0 – October, 2012 (published)

OAuth1 vs OAuth2



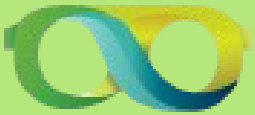
OAuth 1

- Supports only Desktop Application
- Need of Cryptographic signatures
- Complicated – Parsing, Sorting, Encoding
- Access Token valid for years
- Two Security Token for each API call

OAuth 2

- Support for non browser based Application
- No need of Cryptographic signature
- Much less complicated
- Access Token short lived – Refresh Token
- One Security Token for each API call

OAuth2 Roles



- Resource Owner

  Entity capable of granting access to a protected resource

- Client Application

  Application making protected resource requests on behalf of the resource owner

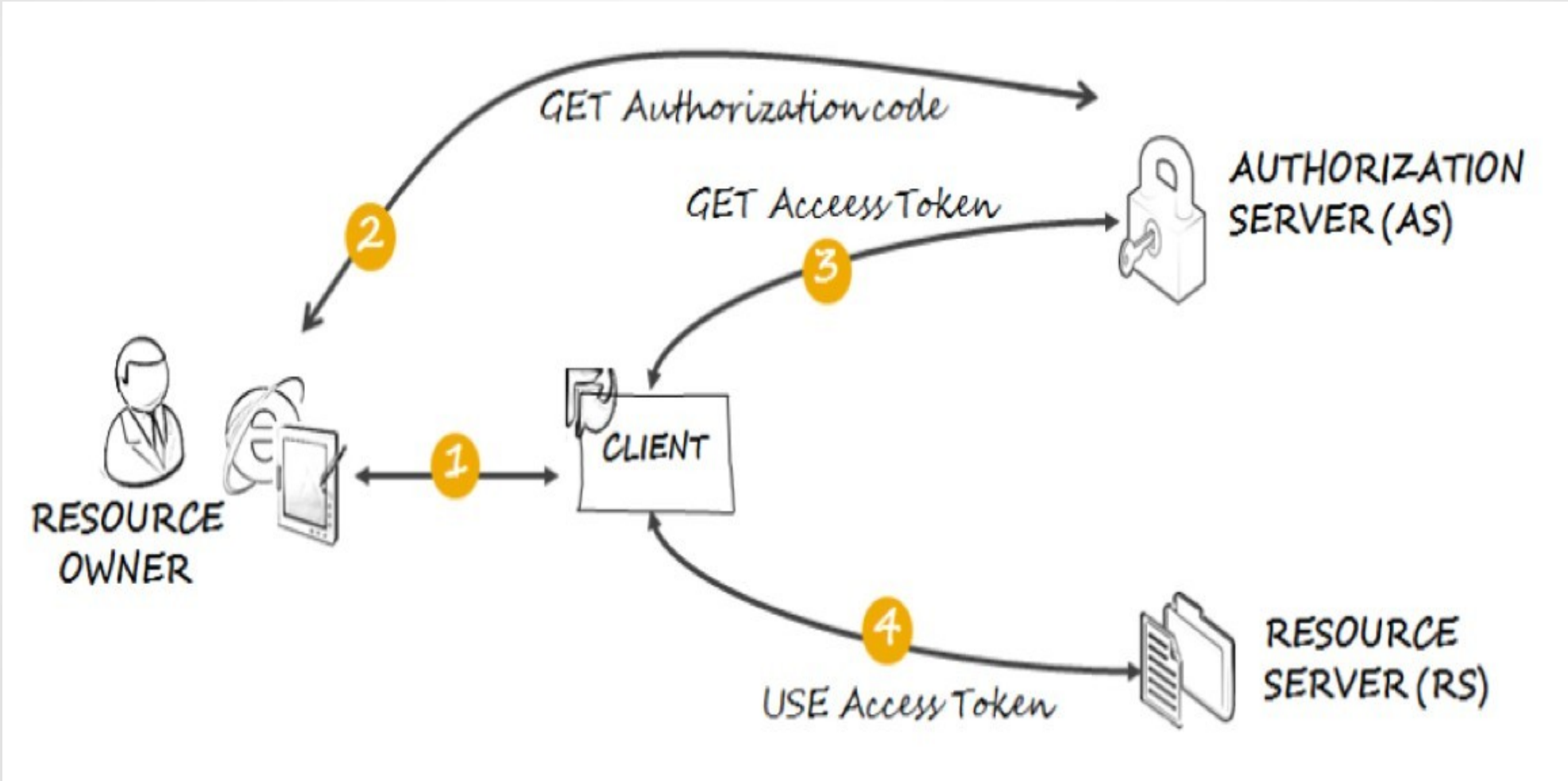
- Resource Server

  The server hosting the protected resources

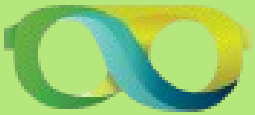
- Authorization Server

  The server issuing access tokens to the clients

Oauth2 Basic Flow



OAuth2 Grant Types

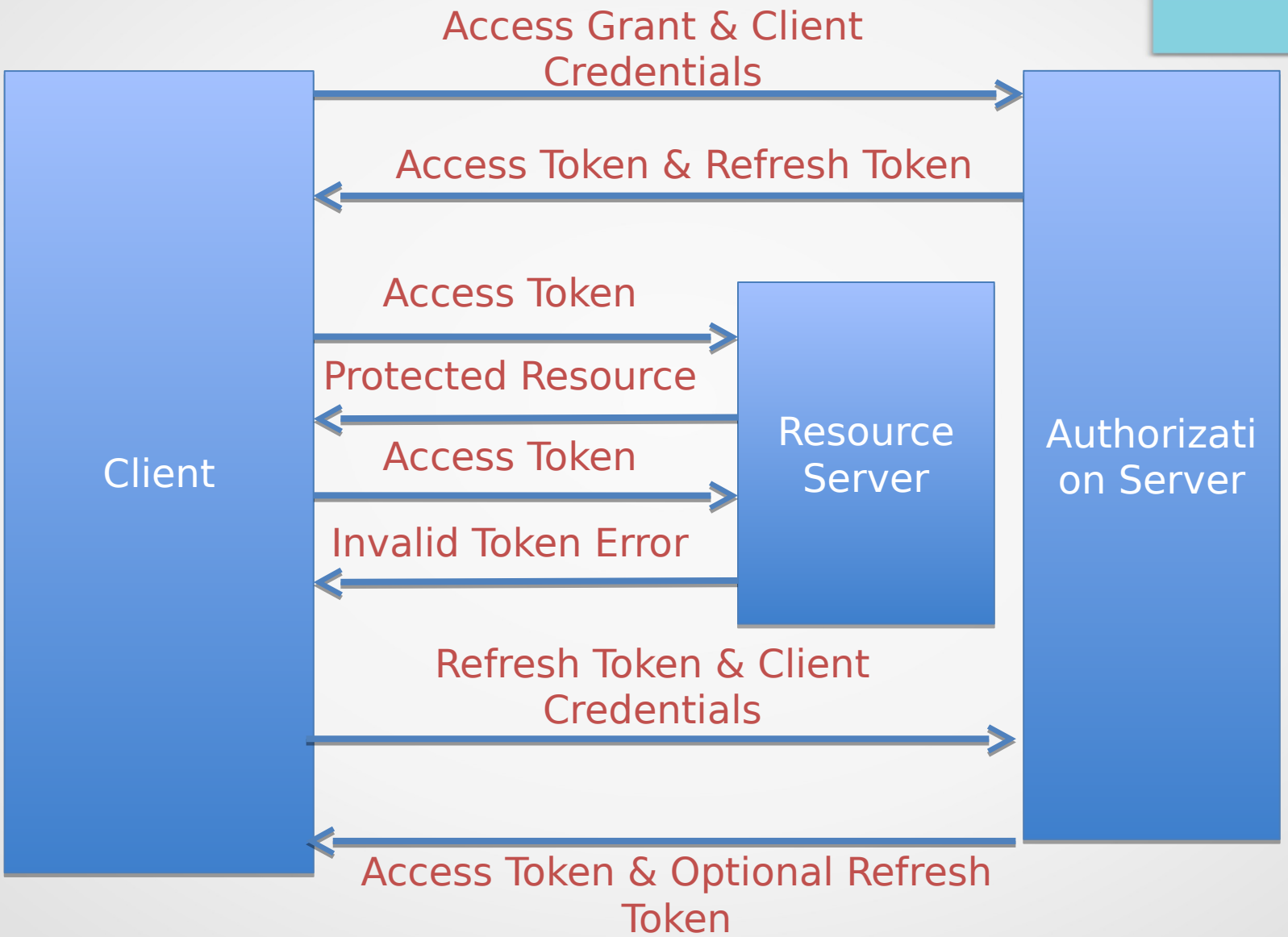


- Authorization Code **(web apps)**
  - Optimized for confidential clients
  - Uses a authorization code from the server
- Implicit **(browser-based and mobile apps)**
  - Optimized for script heavy web apps
  - User can see the access token
- Resource Owner Password Credentials **(user / password)**
  - Used in cases where the user trusts the client
  - Exposes user credentials to the client
- Client Credentials **(application)**
  - Clients gets an access token based on client credentials only

Authorization Code Grant

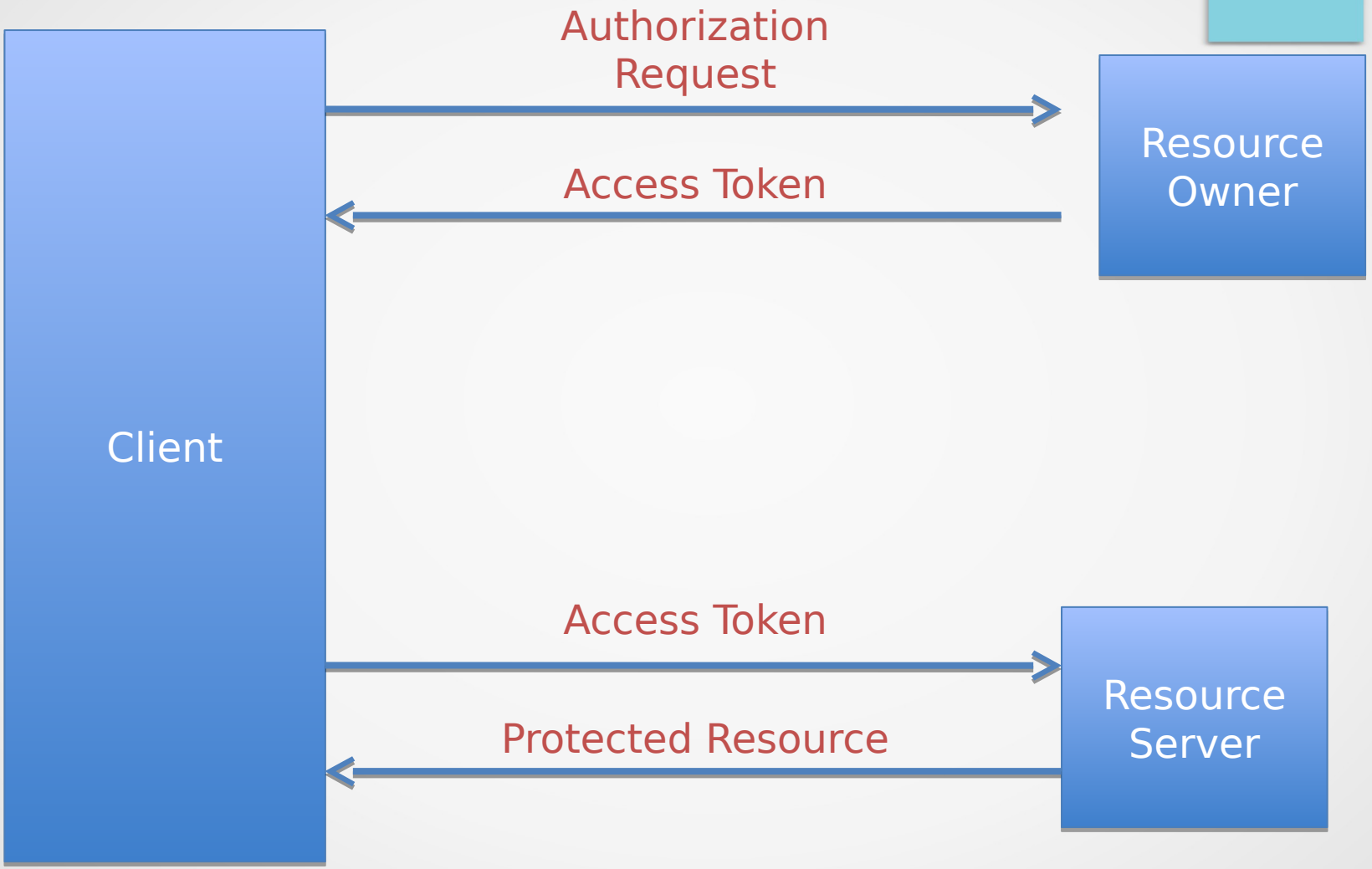
Protocol Flow

Authorization Code Grant (With Refresh Token)



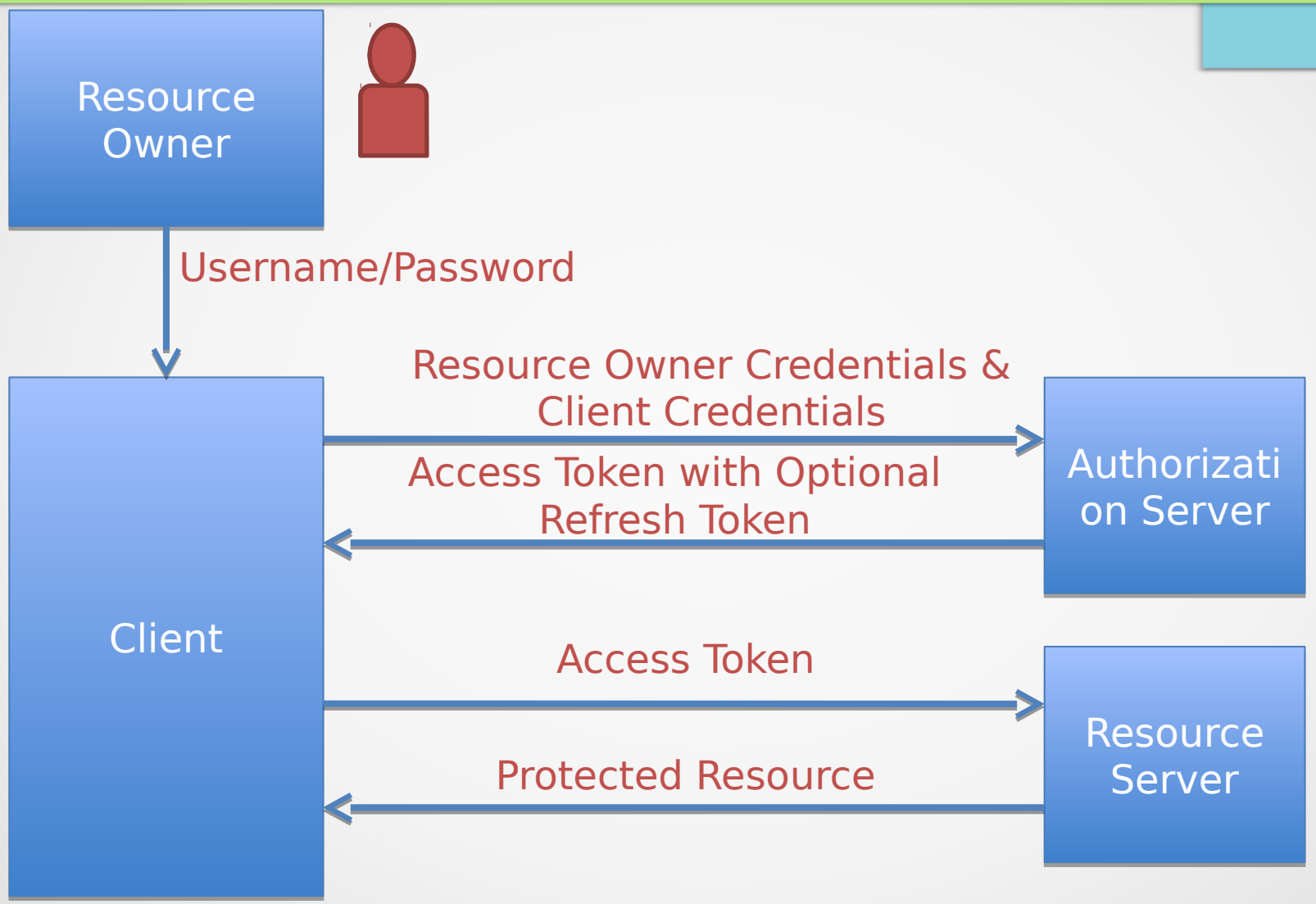
Protocol Flow

Implicit Grant



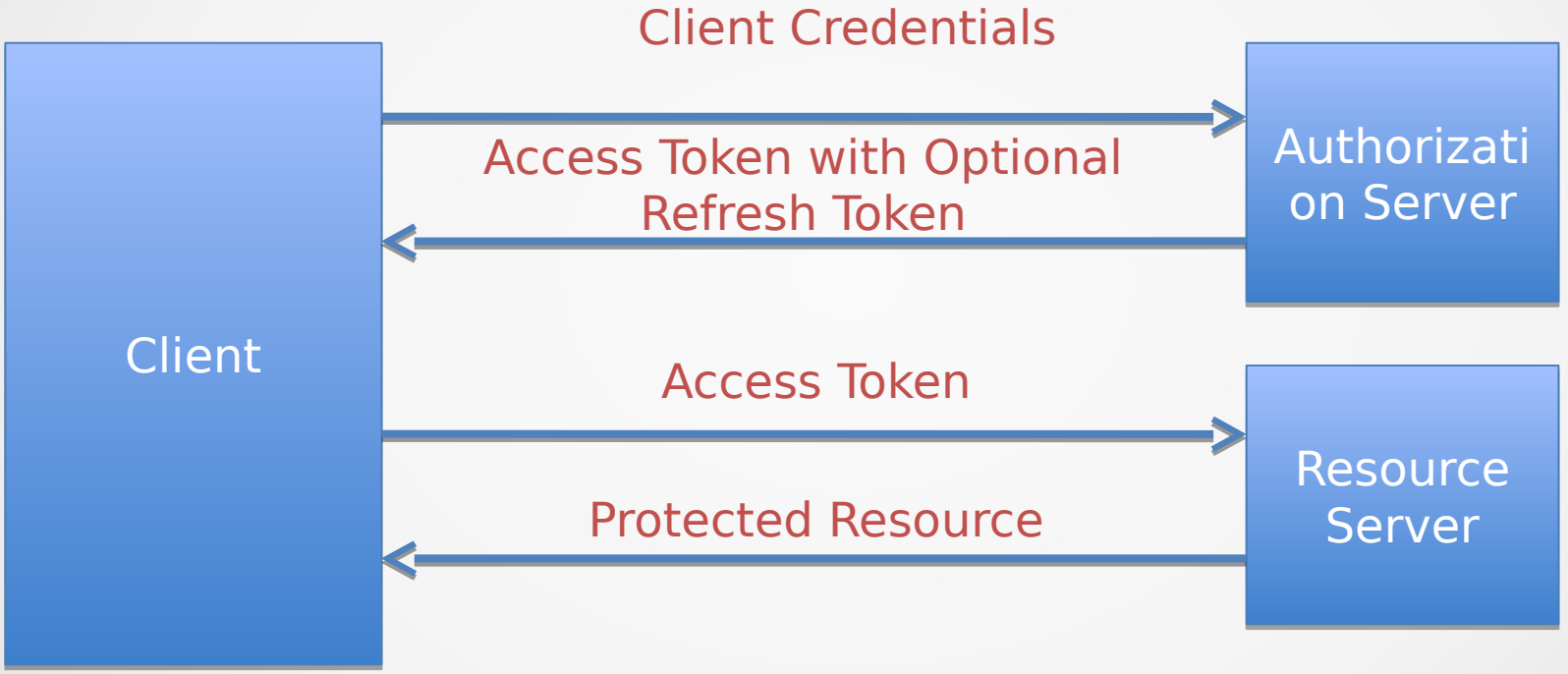
Protocol Flow

Resource Owner Password



Protocol Flow

Client Credentials



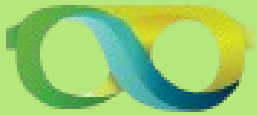
Protocol Flow

OAuth2 Tokens

- Types
  - Bearer
    - Large random token
    - Need SSL to protect it in transit
    - Server needs to store it securely hashed like a user password
  - Mac
    - Uses a nonce to prevent replay
    - Does not required SSL
    - OAuth 1.0 only supported
- Access Token
  - Short-lived token
- Refresh Token
  - Long-lived token

```
{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":“bearer",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
  ,
}
```

OAuth2 Pros and Cons



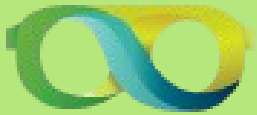
Pros

- Integration of third party apps to any sites
- Access can be granted for limited scope or duration
- No need for users to give password on third party site

Cons

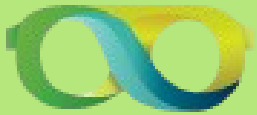
- Writing an authorization server is somewhat complex
- Interoperability issues and HTTP protocol only
- Bad implementations can be security issues

Spring Security OAuth



- OAuth1.0a and OAuth2.0 Support
- Supports OAuth2 full features
- Authorization Server
- Resource Server
- Client
- Integration with Spring MVC
- Configuration using annotations

Spring Authorization Server



- @EnableAuthorizationServer

  Annotation

- ClientDetailsServiceConfigurer

  Defines Client Details Services

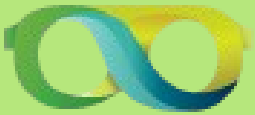
- AuthorizationServerTokenServices

  Manage OAuth2 tokens

- AuthorizationServerEndpointConfigurer

  Grant types support

Spring Resource Server



- Can be same or Deployed in seperate Server
- @EnableResourceServer

  Annotation

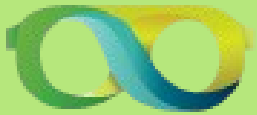
- Expression Based access control

  #oauth2.clientHasRole

  #oauth2.clientHasAnyRole

  #oauth2.denyClient

Spring OAuth2 Client



- Manages the redirection to and from the OAuth authentication URI

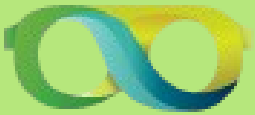
- @EnableOAuth2Client

  Annotation

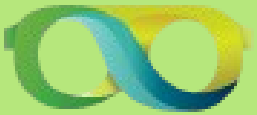
- OAuth2RestTemplate

  Wrapper client object to access the resources

References



- http://oauth.net/2/
- http://tools.ietf.org/html/rfc6749
- http://projects.spring.io/spring-security-oauth/
- https://github.com/spring-projects/spring- security-oauth



Thank You !