Login & Access Control in Angular

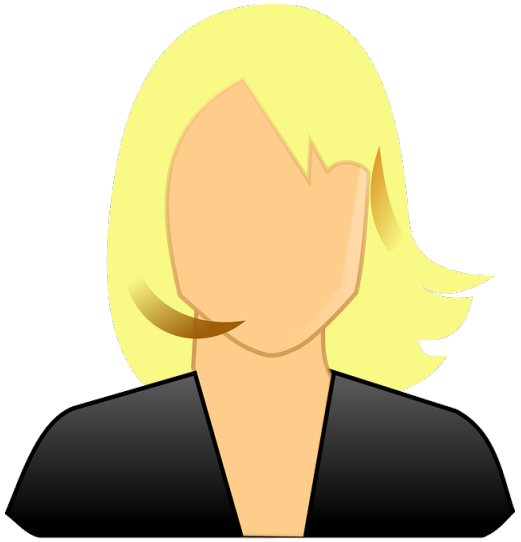
Alex Thalhammer

Contents

- Motivation
- OAuth 2
- Single Sign on and OpenId Connect

Motivation

Access to App and Backend



Requirements for Modern Apps

Service delegates to other services	Cross Origin Requests	Using existing Identity Solutions
Loosely Coupling to Identity Solution	Single Sign on/ out	Protect from XSRF

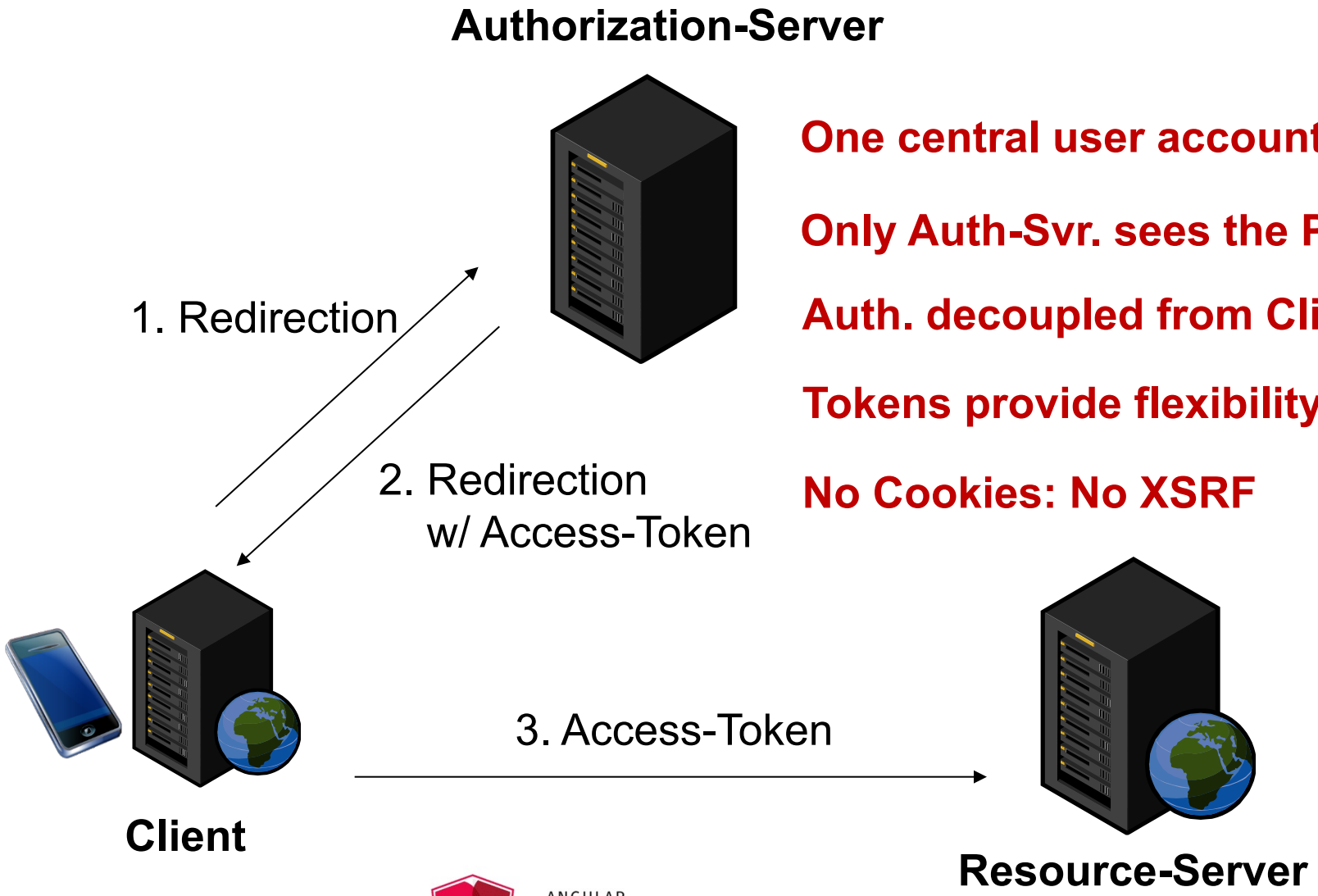
Roles

Authorization-Server

Client

Resource-Server

Flow



One central user account

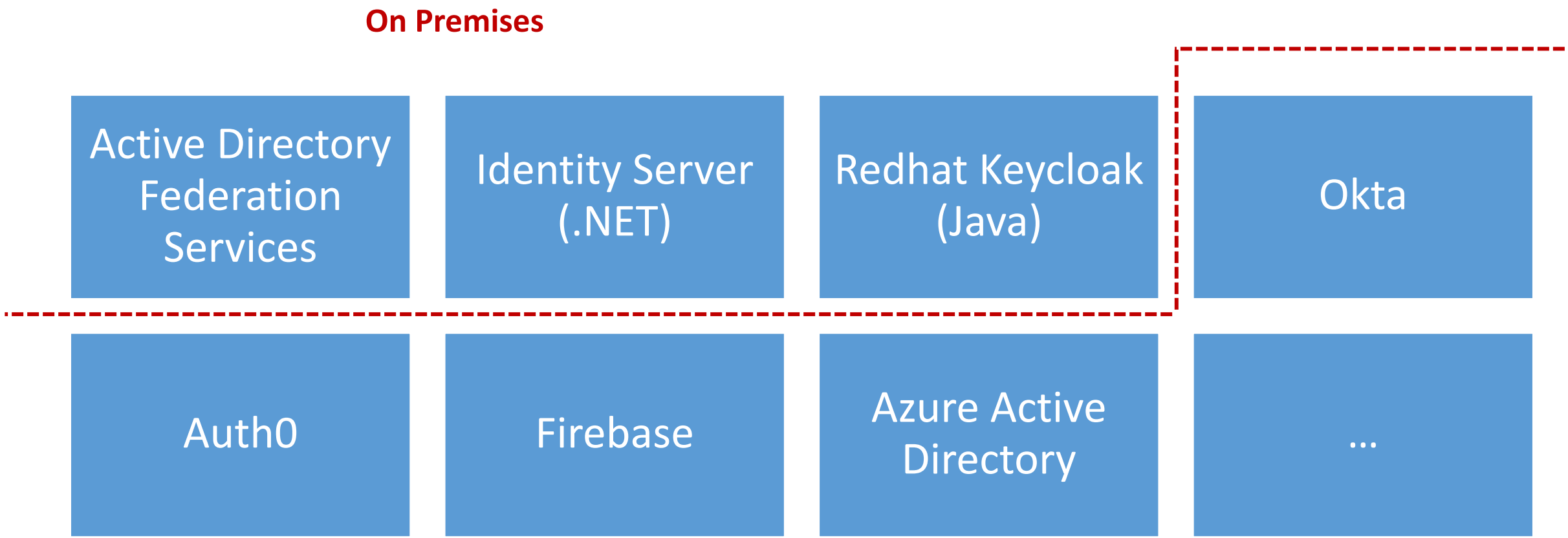
Only Auth-Svr. sees the Password

Auth. decoupled from Client

Tokens provide flexibility

No Cookies: No XSRF

Lots of Auth Server out there …

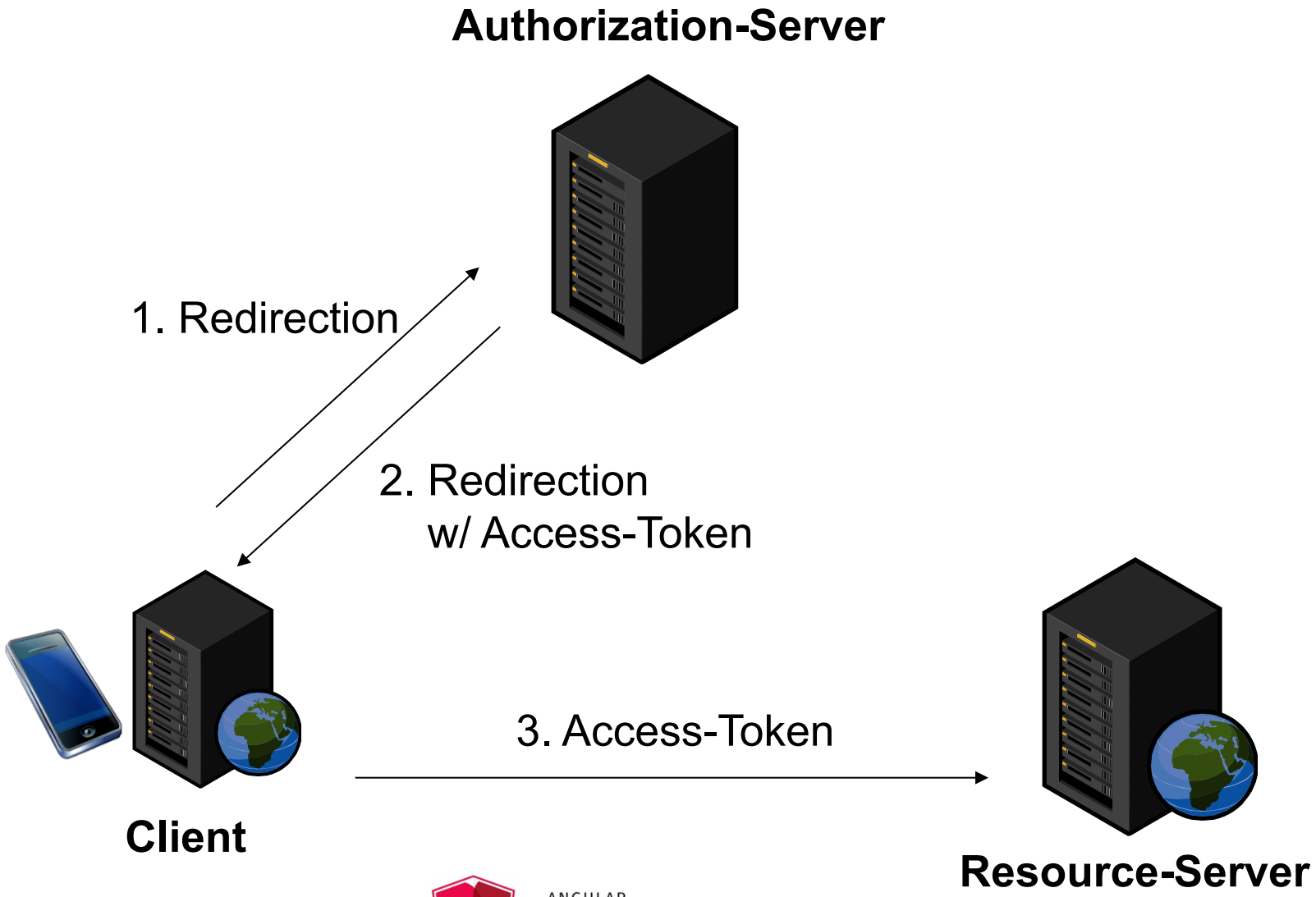


OAuth 2 and OpenId Connect

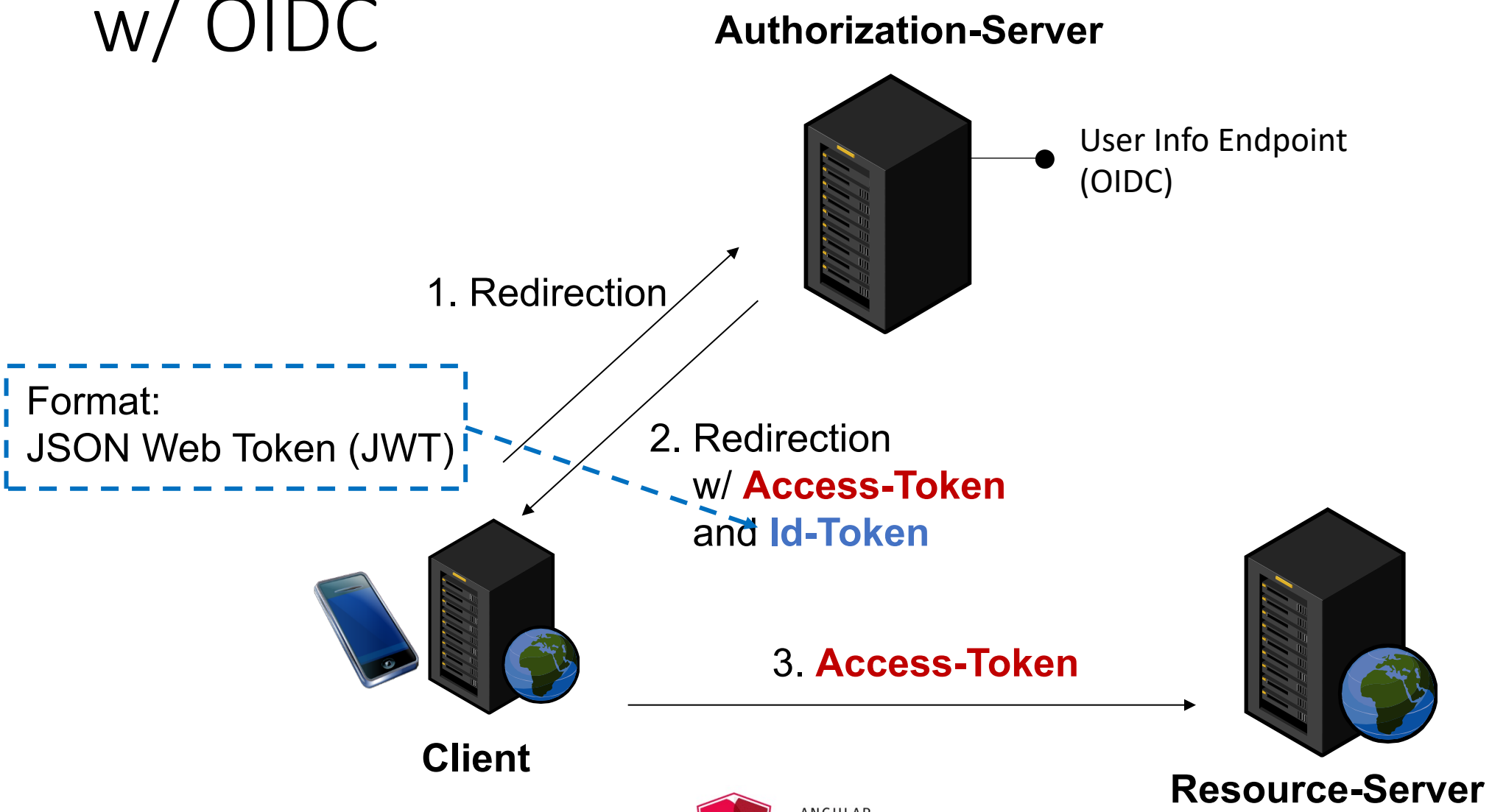
What is OAuth 2?

- Developed by Twitter and Ma.gnolia
- Protocol to delegate restricted rights
- Used by Companies like Google, Facebook, Flickr, Microsoft, Salesforce.com or Yahoo!
- Several Flows for different use cases
- Leverages HTTPS!

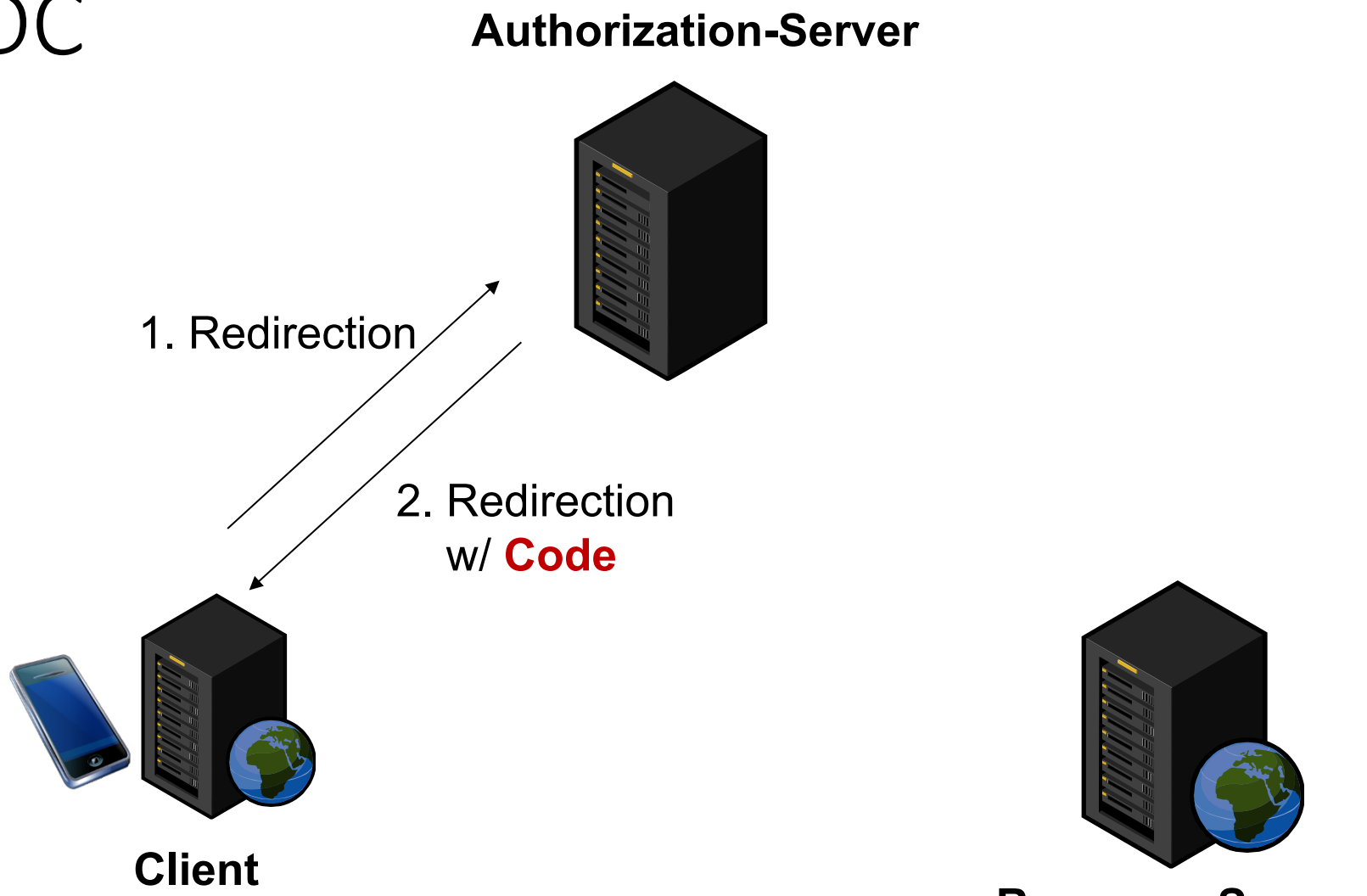
Implicit Flow for SPA



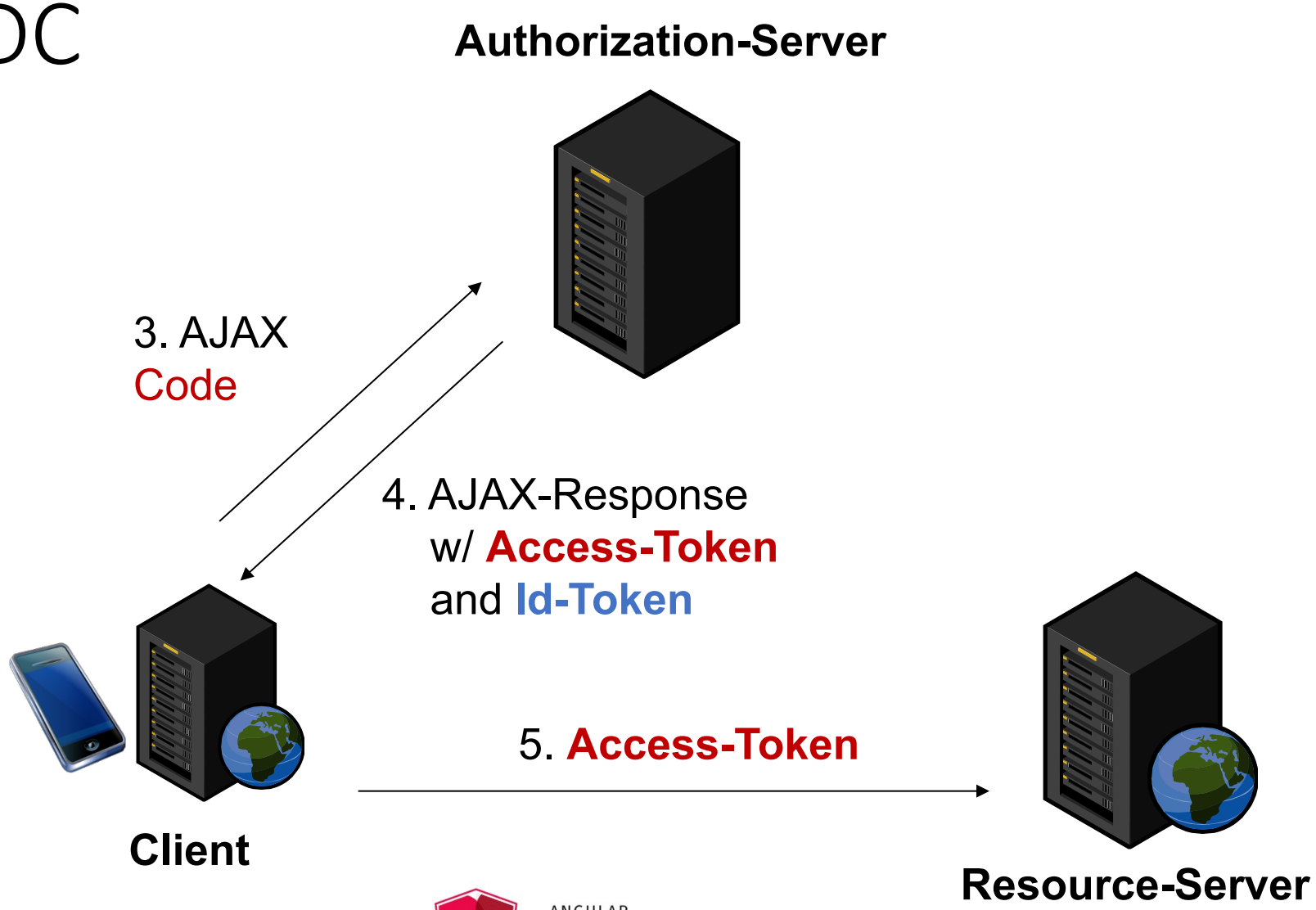
Implicit Flow w/ OIDC



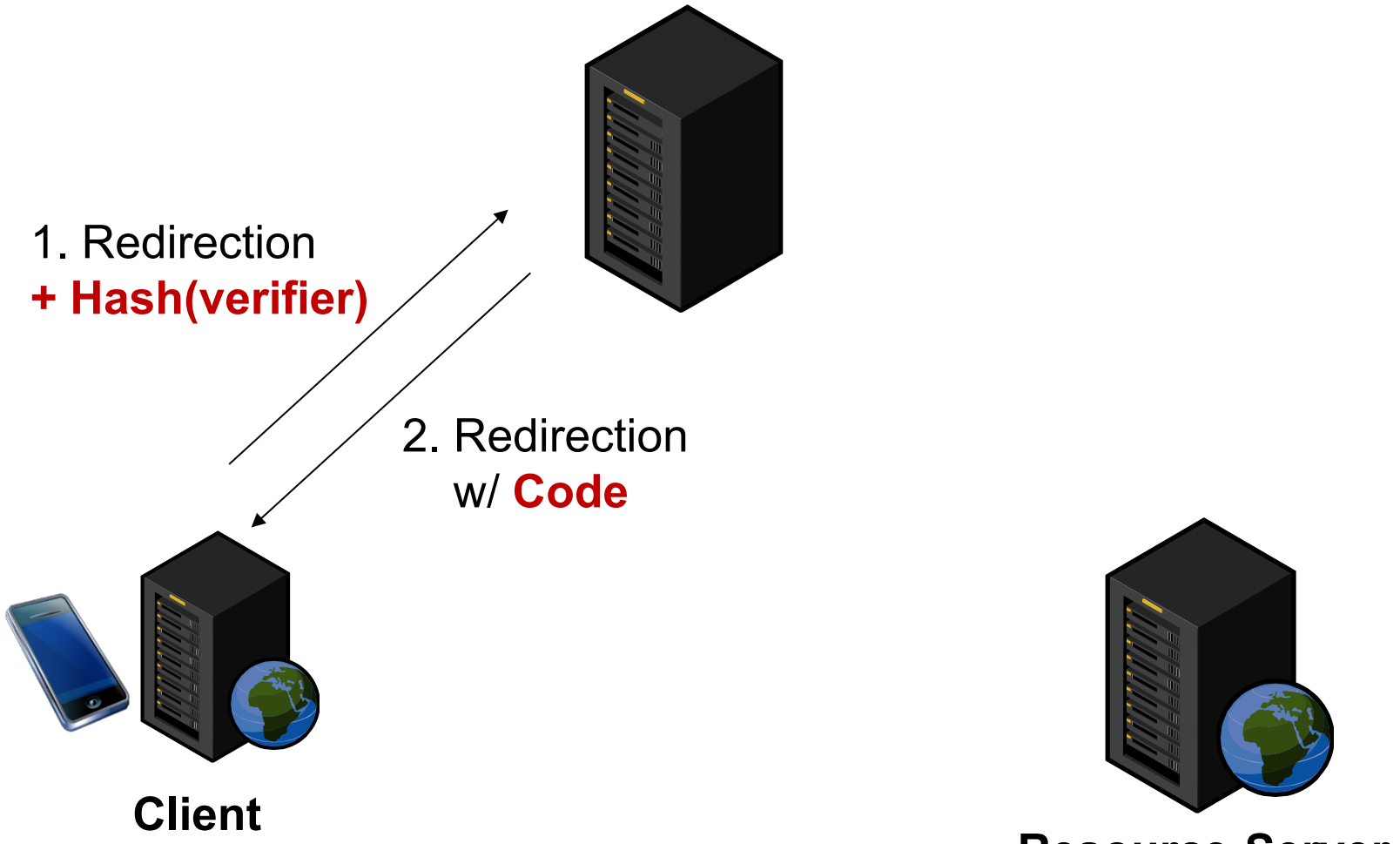
Code Flow w/ OIDC



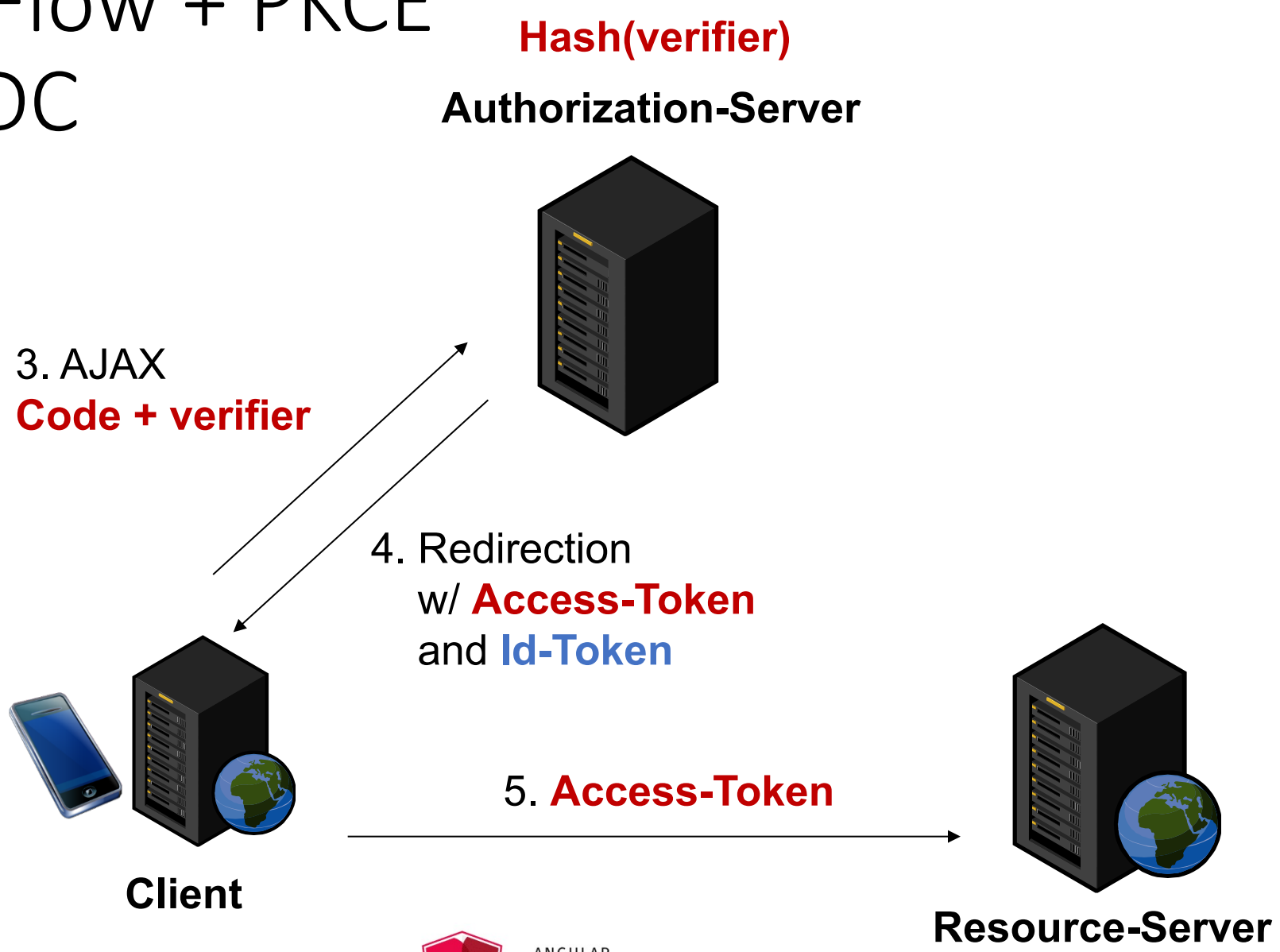
Code Flow w/ OIDC



Code Flow + PKCE w/ OIDC



Code Flow + PKCE w/ OIDC



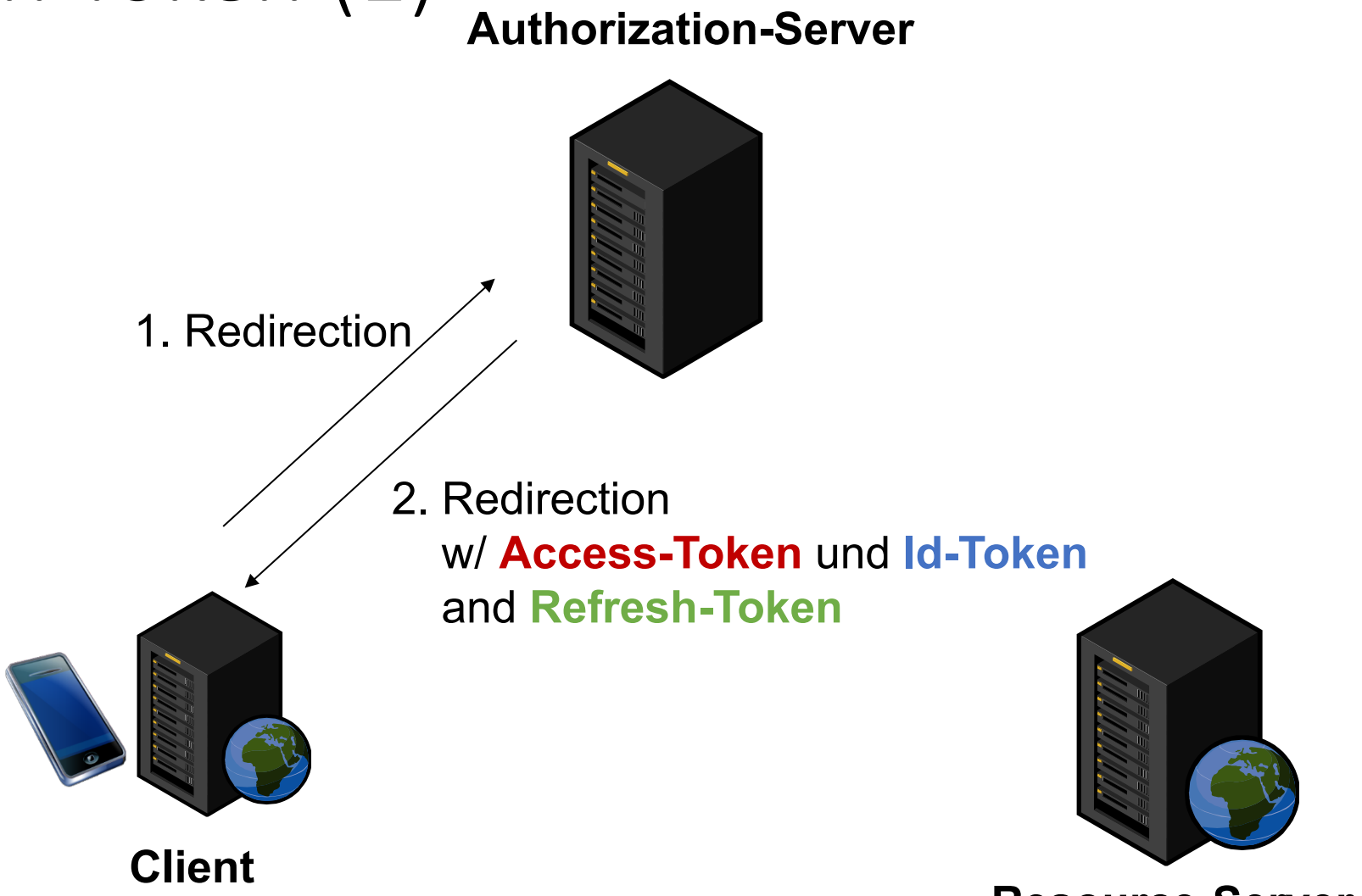
Token Refresh

Why Token Refresh?

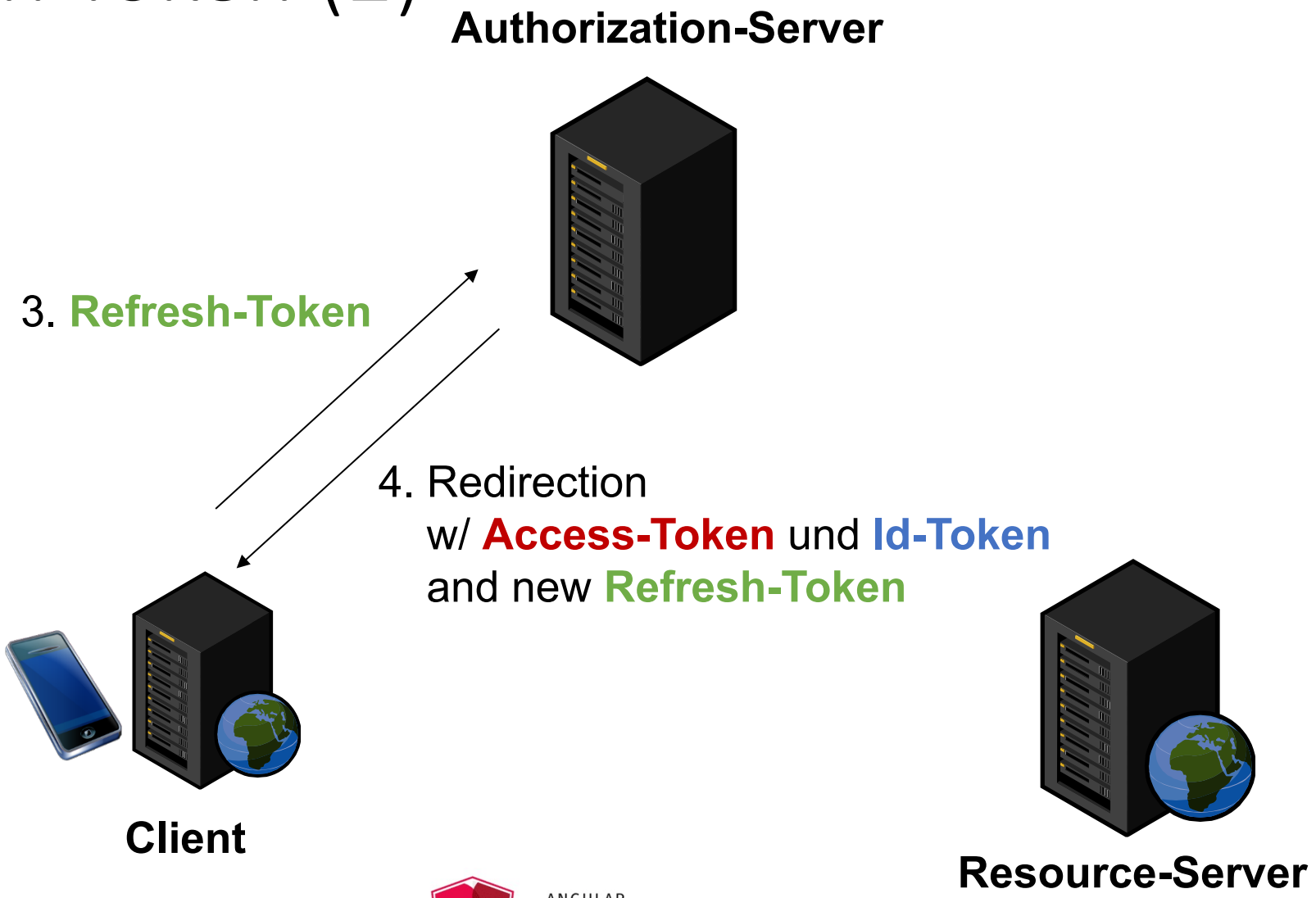
Short living Tokens increase Security

Users don't want to login over and over again

Refresh Token (1)



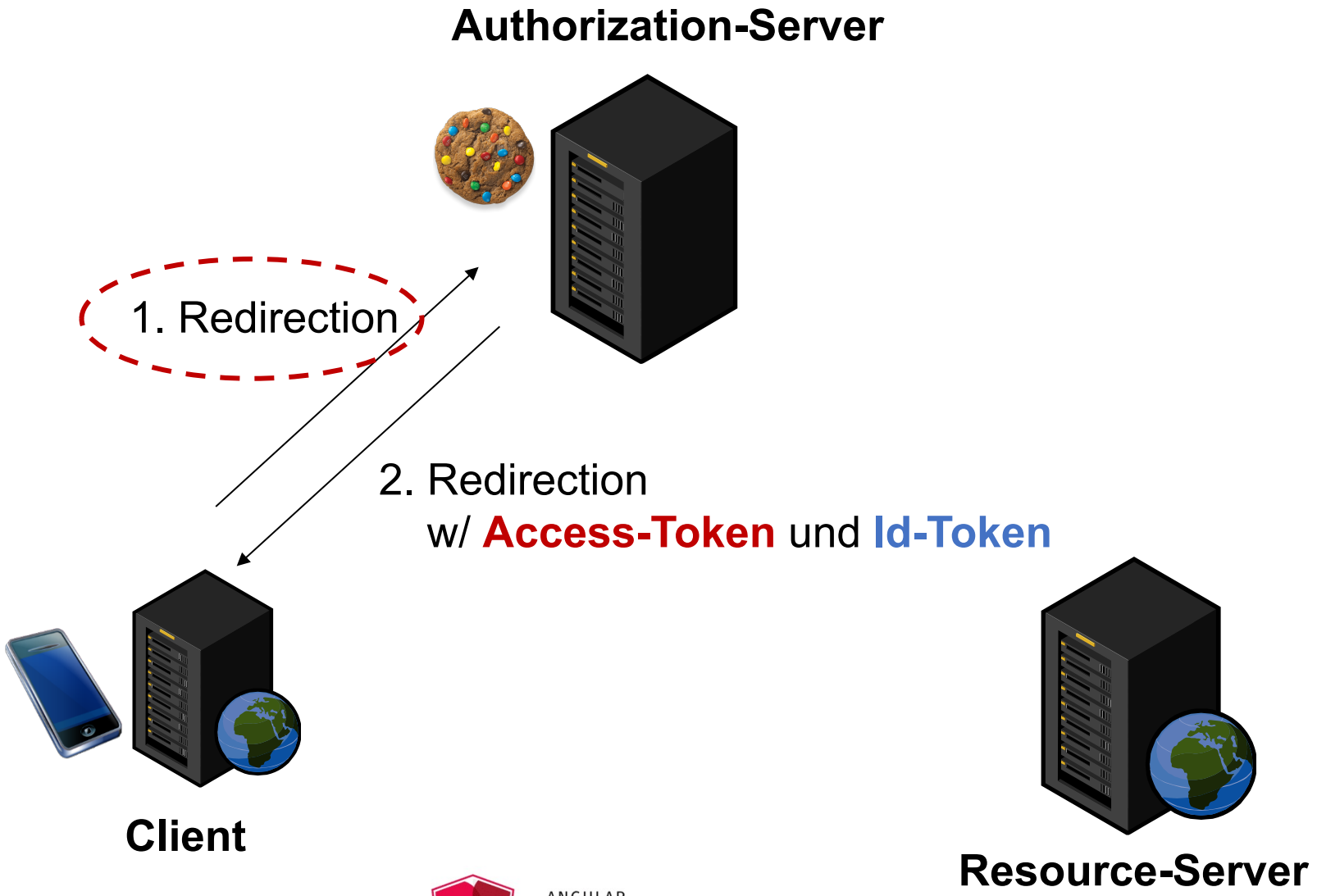
Refresh Token (2)



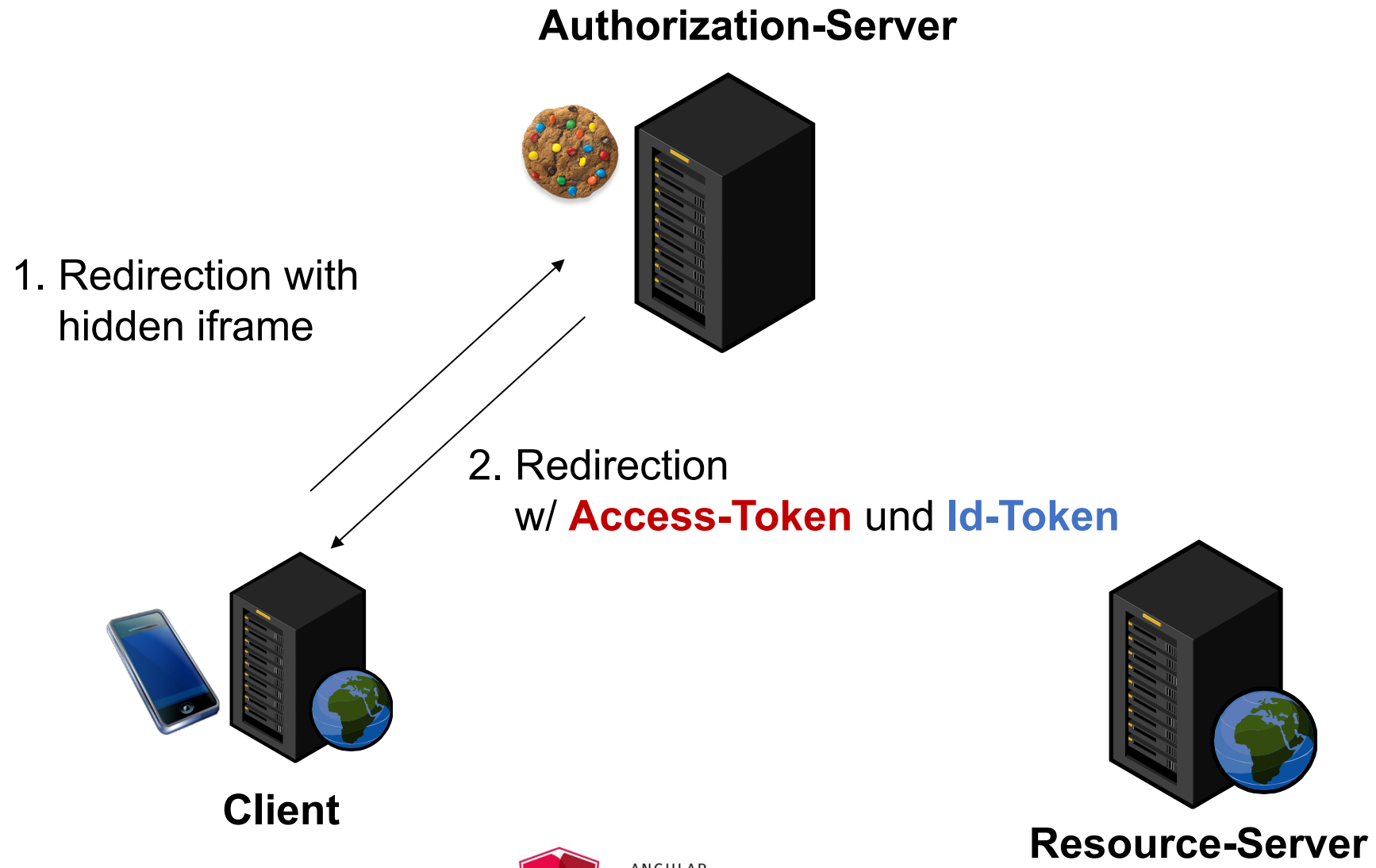
Refresh-Token and Browsers

- OAuth 2.0 Security Best Current Practice allows it under specific circumstances
- Security Audit (XSS!)
- Refresh Token needs to be one-time token
- After Refresh: Client gets new refresh token
- If used by several users: log out both
- Bind Refresh Token to session (invalidate it during logout)

Alternative: Refresh w/ Cookie



Alternative: Silent Refresh



Lab

Conclusion

Token: Flexibility, Cross Origin …

OAuth 2: Access to Service

OpenId Connect: SSO at Client

Implicit Flow

Code Flow + PKCE