Login & Access Control in Angular

Alex Thalhammer

Outline

- Motivation
- OAuth 2 and OpenId Connect
- Demo & Labs
- Refresh Token

Motivation

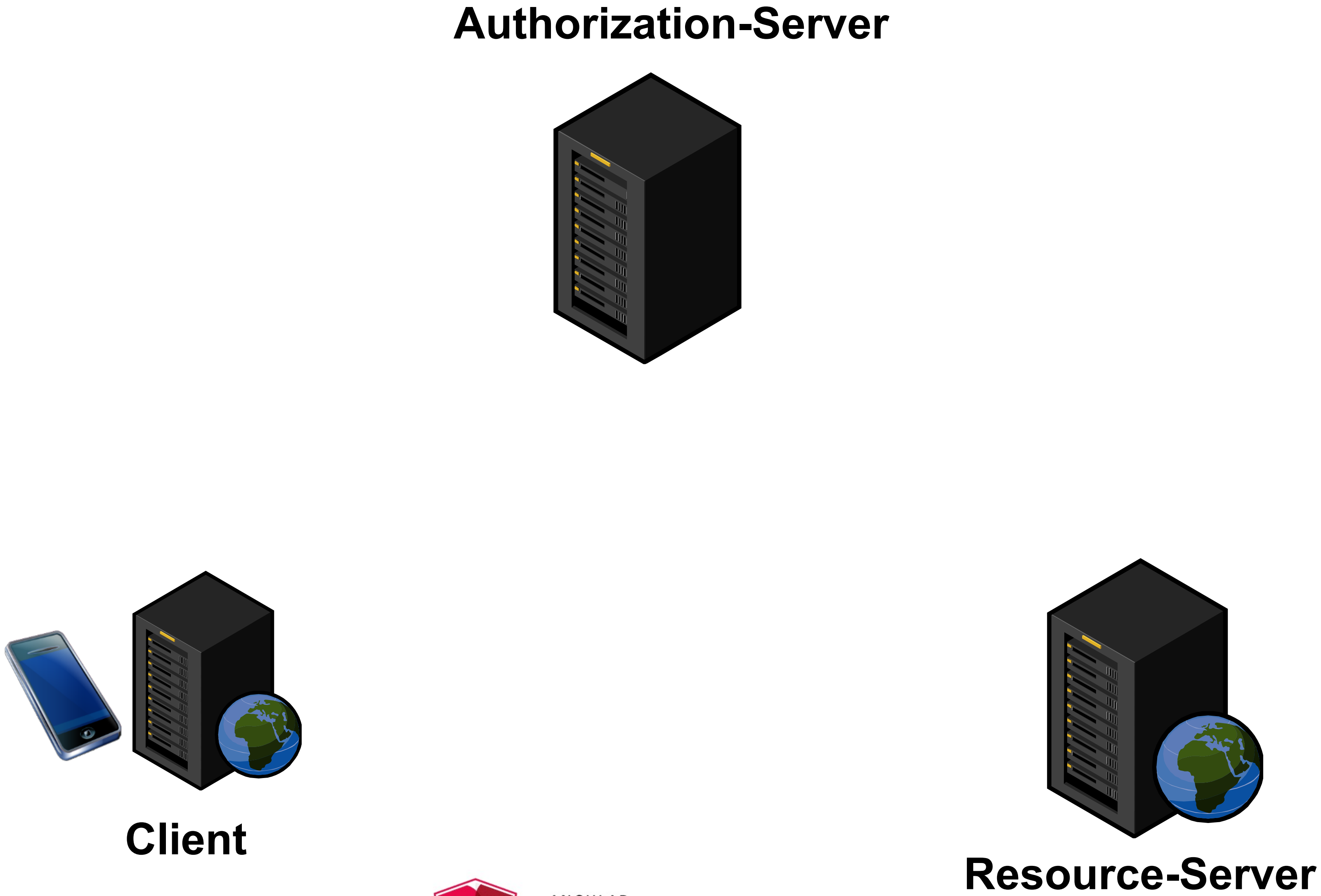
Access to App and Backend

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

PASSPORT
E PLURIBUS
UNUM

Roles

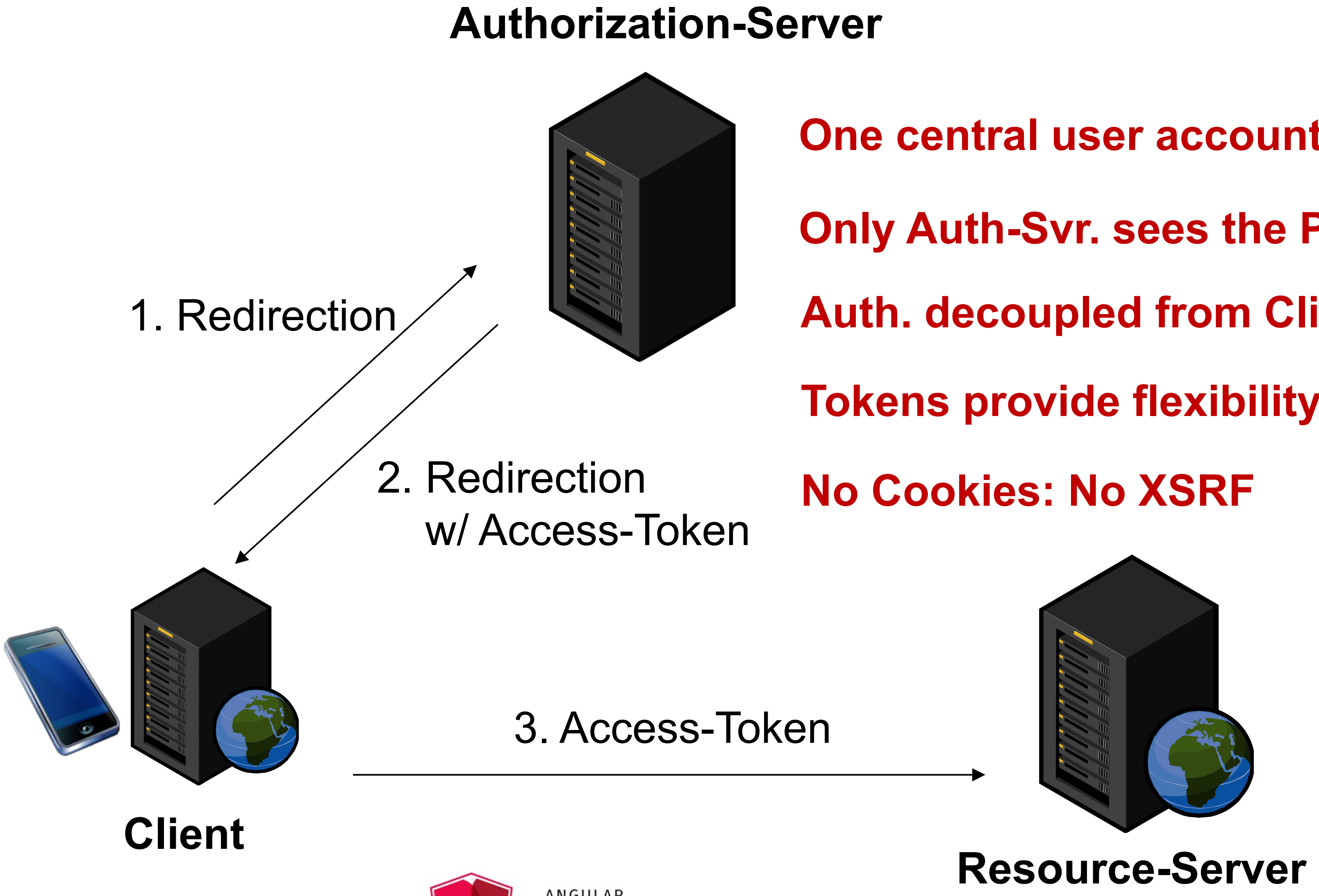


Authorization-Server

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

OAuth 2 and OpenId Connect

OAuth 2 and OpenId Connect

- 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!

DEMO

Lab

Token Refresh

Why Token Refresh?

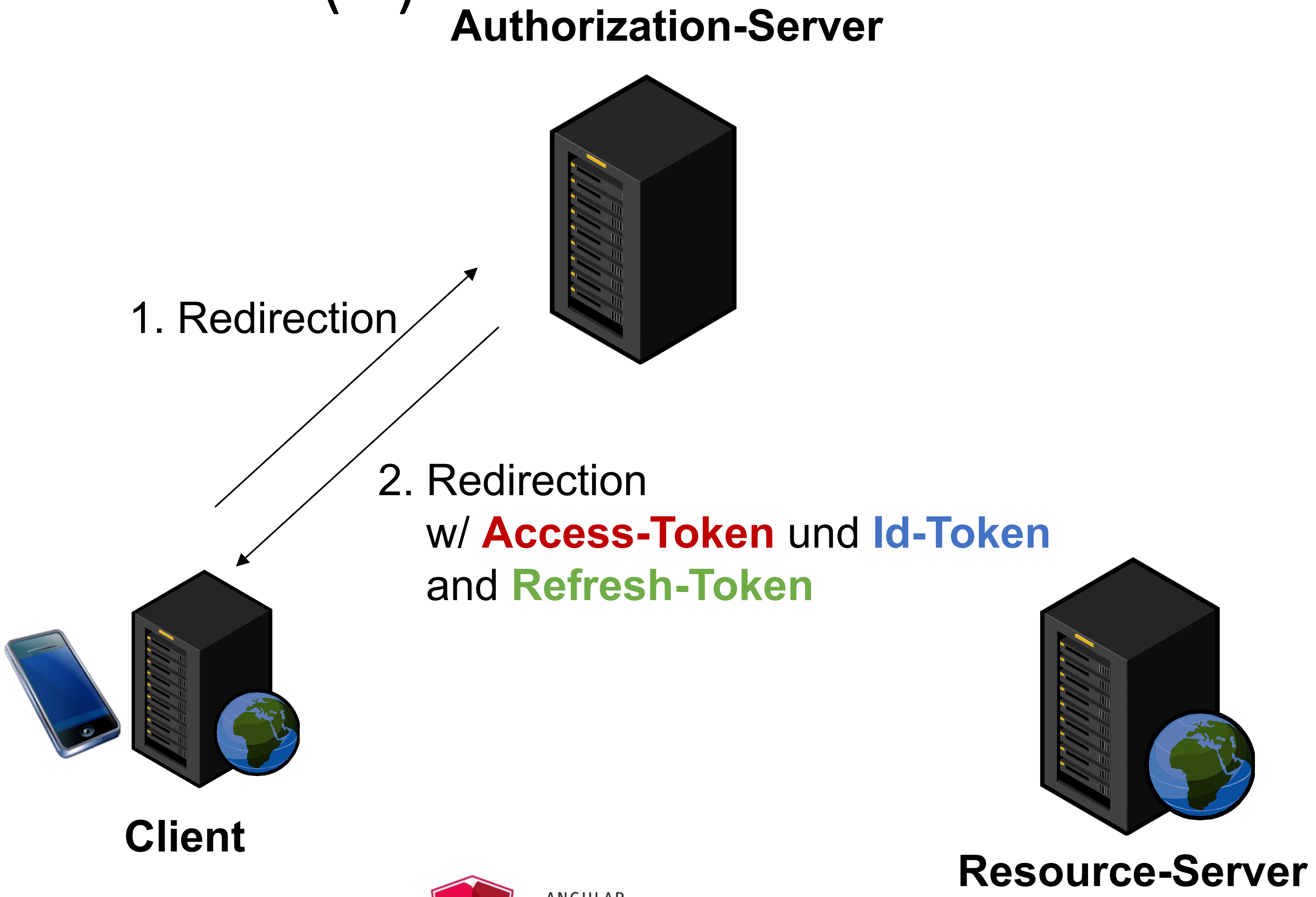
Short living Tokens increase Security

Users don't want to login over and over again

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)

Refresh Token (1)



Authorization-Server

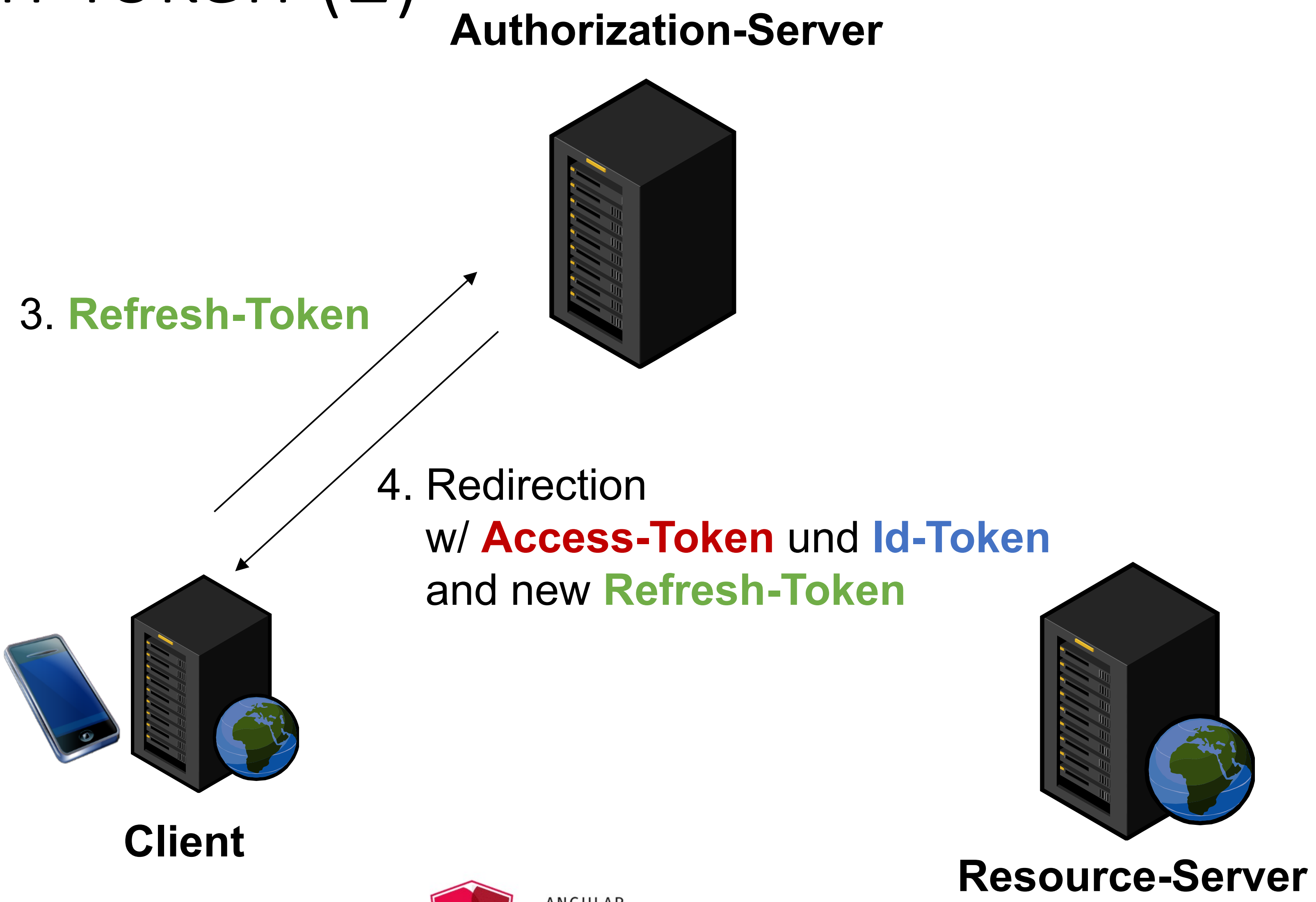
1. Redirection

2. Redirection
w/ Access-Token und Id-Token
and Refresh-Token

Client

Resource-Server

Refresh Token (2)



Authorization-Server

3. **Refresh-Token**

4. Redirection
w/ **Access-Token** und **Id-Token**
and new **Refresh-Token**

Client

Resource-Server

That’s it for basic authentication!

- Questions so far?