ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Login and Access Control in Angular

angular-architects.io

Contents

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

Motivation

Access to App and Backend

Requirements for Modern Apps

Service delegates to other services

Cross Origin Requests

Using existing Identity Solutions

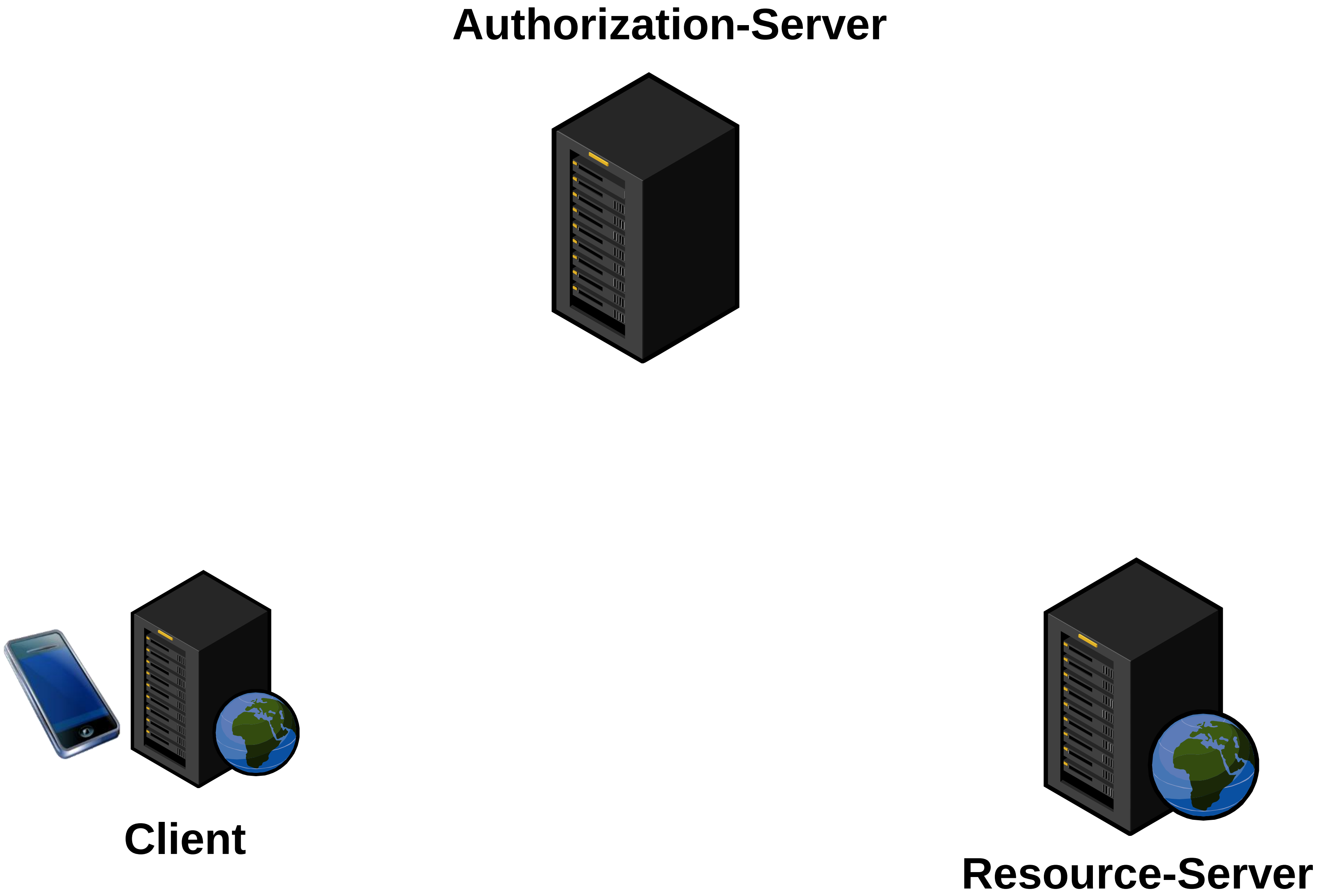
Loosely Coupling to Identity Solution

Single Sign on/ out

Protect from XSRF

E PLURIBUS
UNUM
PASSPORT

Roles

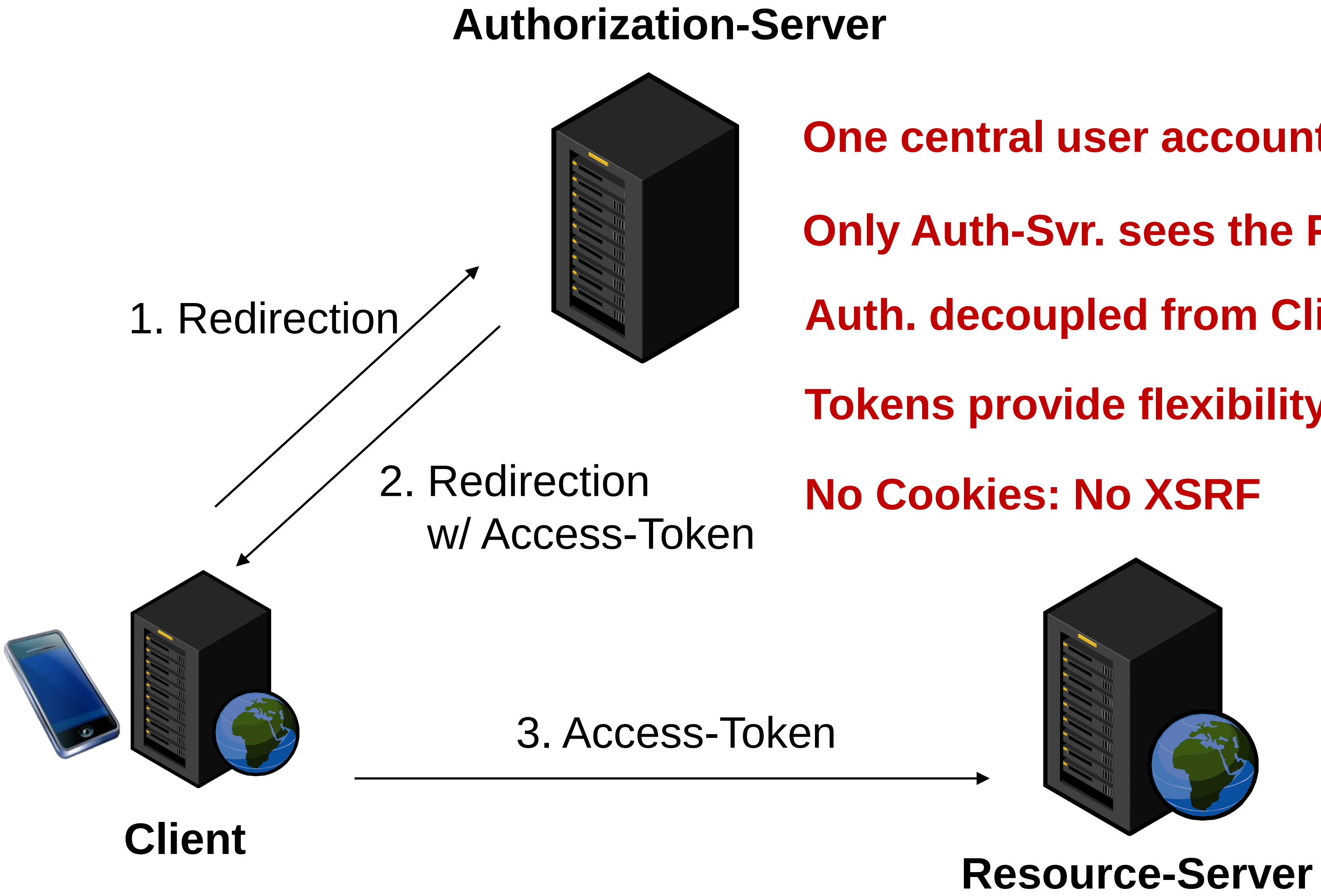


Authorization-Server

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 ...

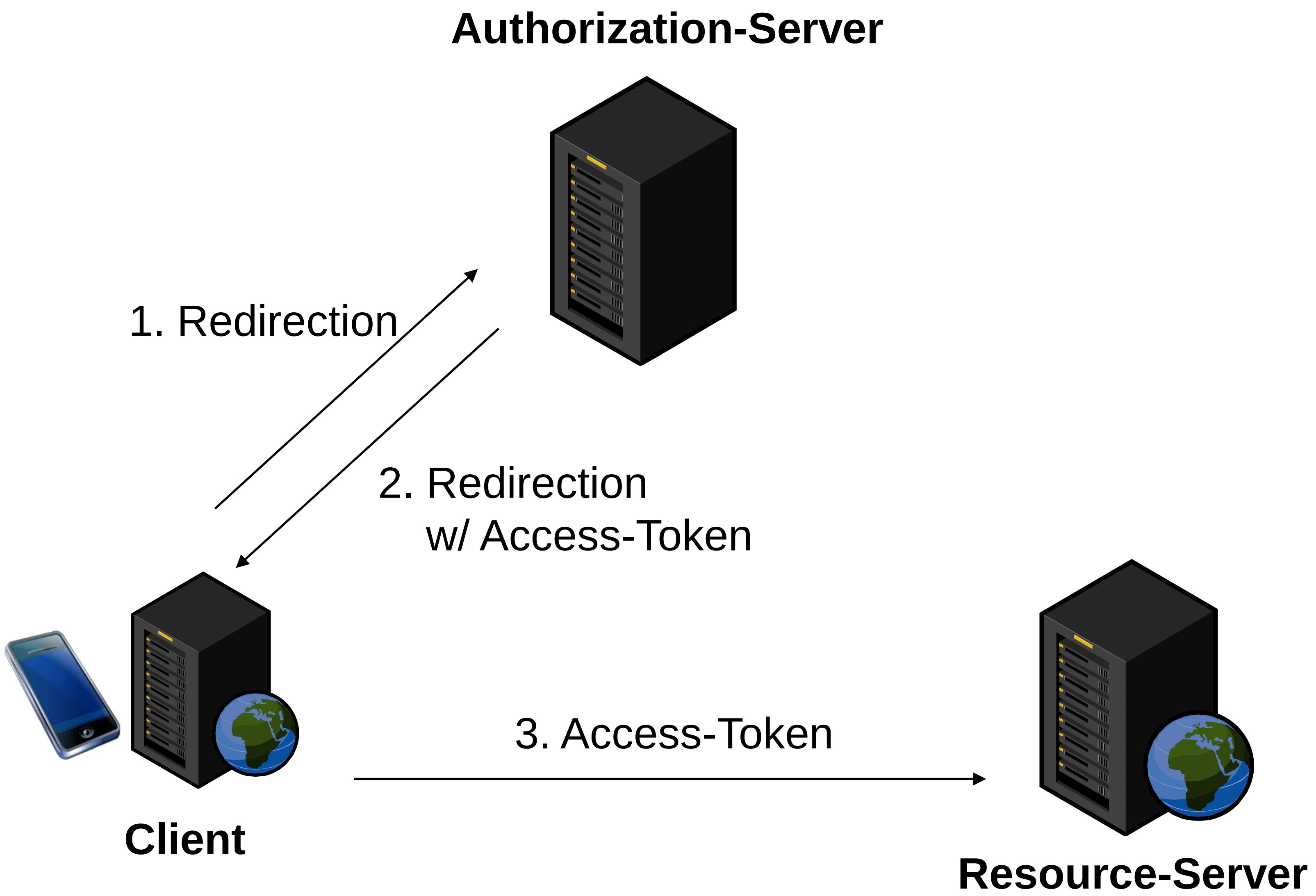
- Active Directory Federation Services
- Identity Server (.NET)
- Redhat Keycloak (Java)
- Okta
- Auth0
- Firebase
- Azure Active Directory
- ...

OAuth 2 and OpenId Connect

What is OAuth 2?

- 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



Authorization-Server

1. Redirection

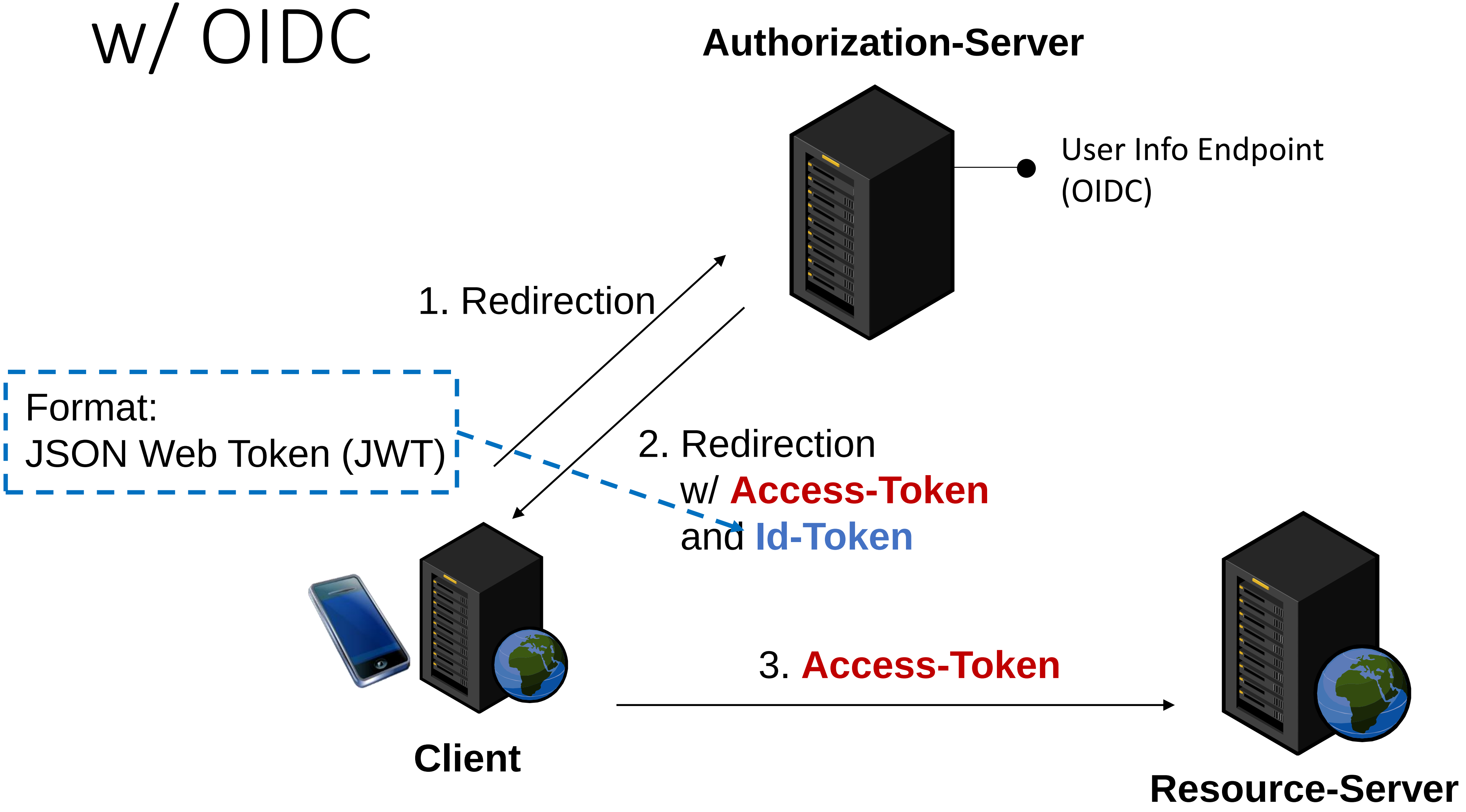
2. Redirection
w/ Access-Token

3. Access-Token

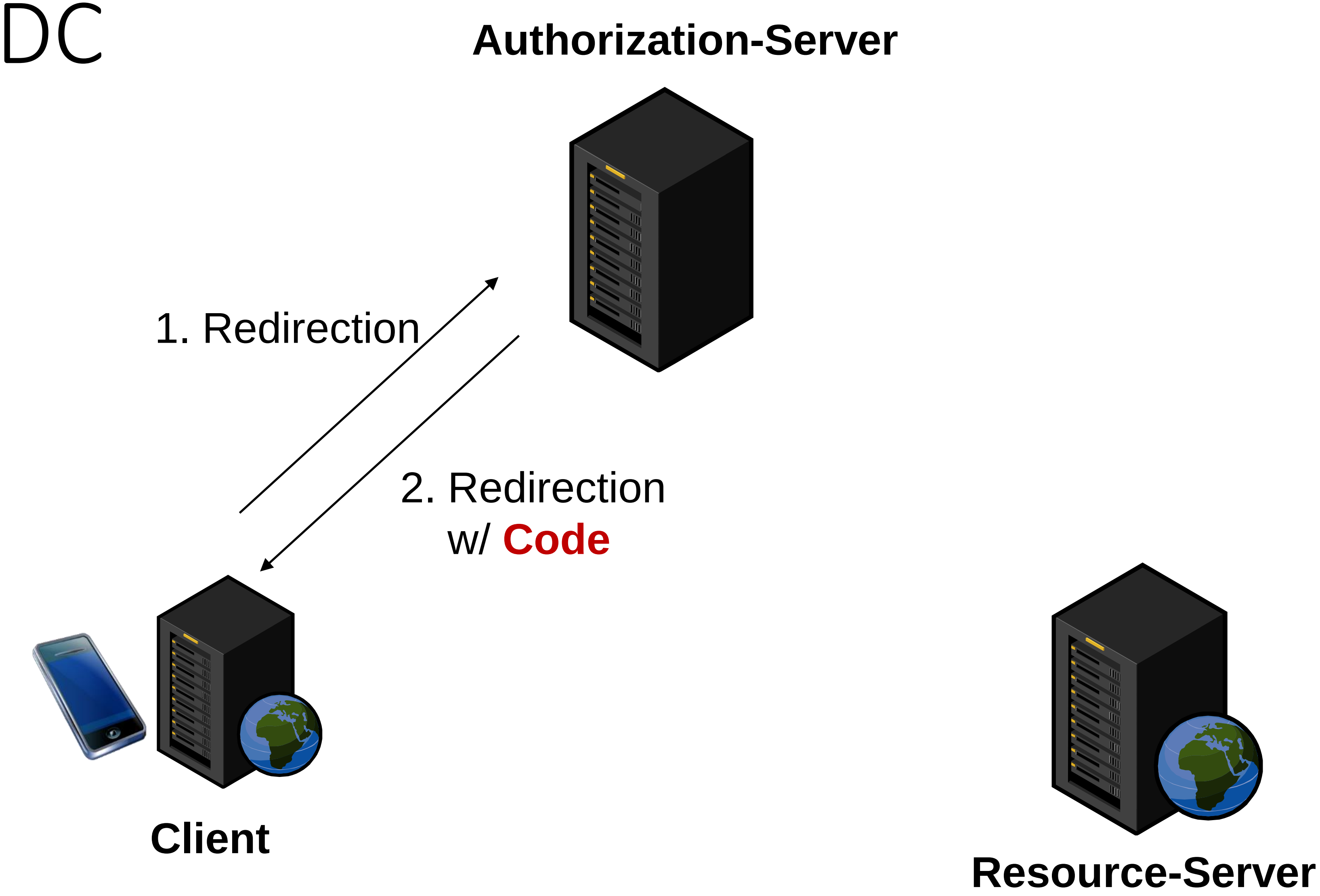
Client

Resource-Server

Implicit Flow w/ OIDC



Code Flow w/ OIDC


Authorization-Server

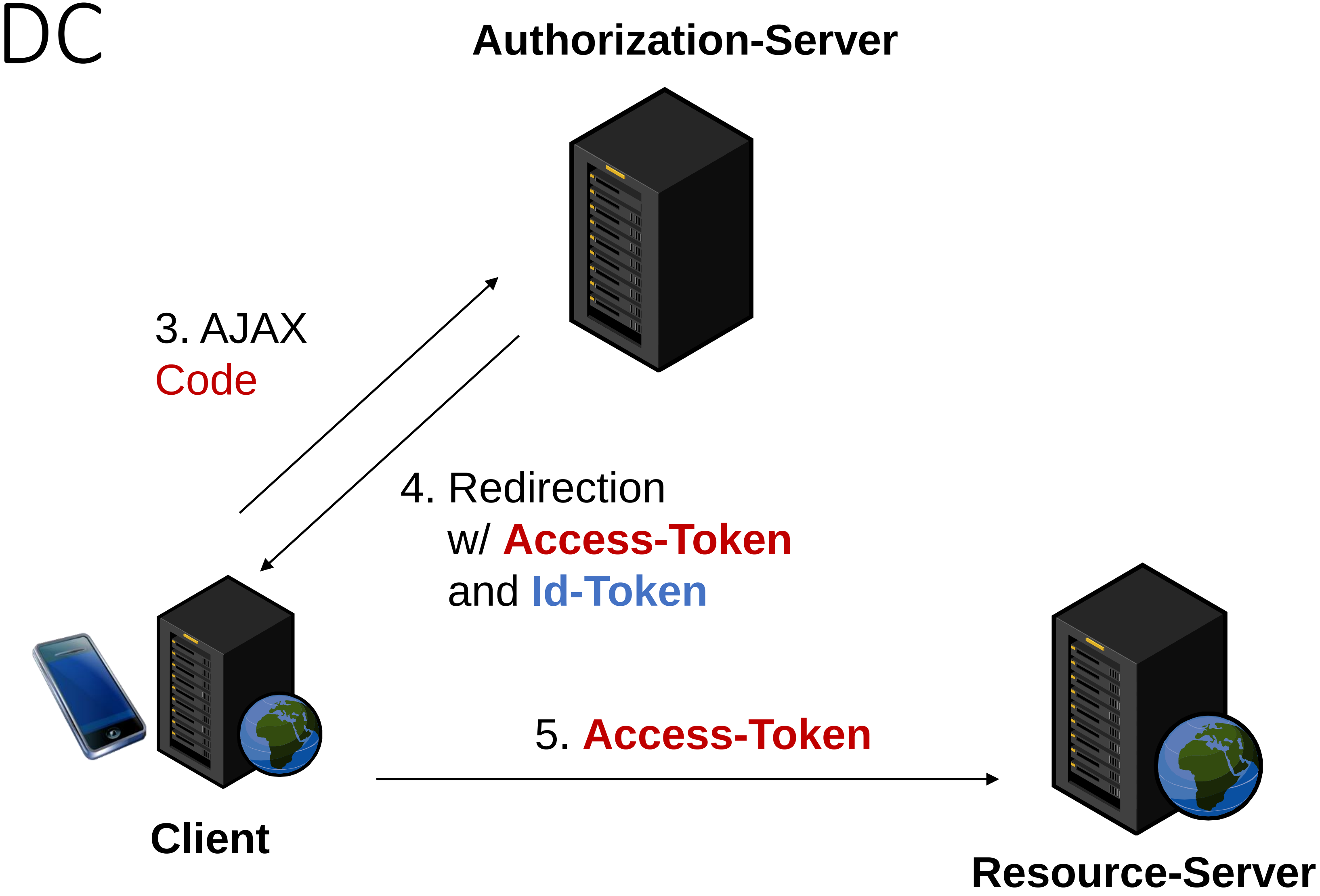
1. Redirection

2. Redirection w/ **Code**

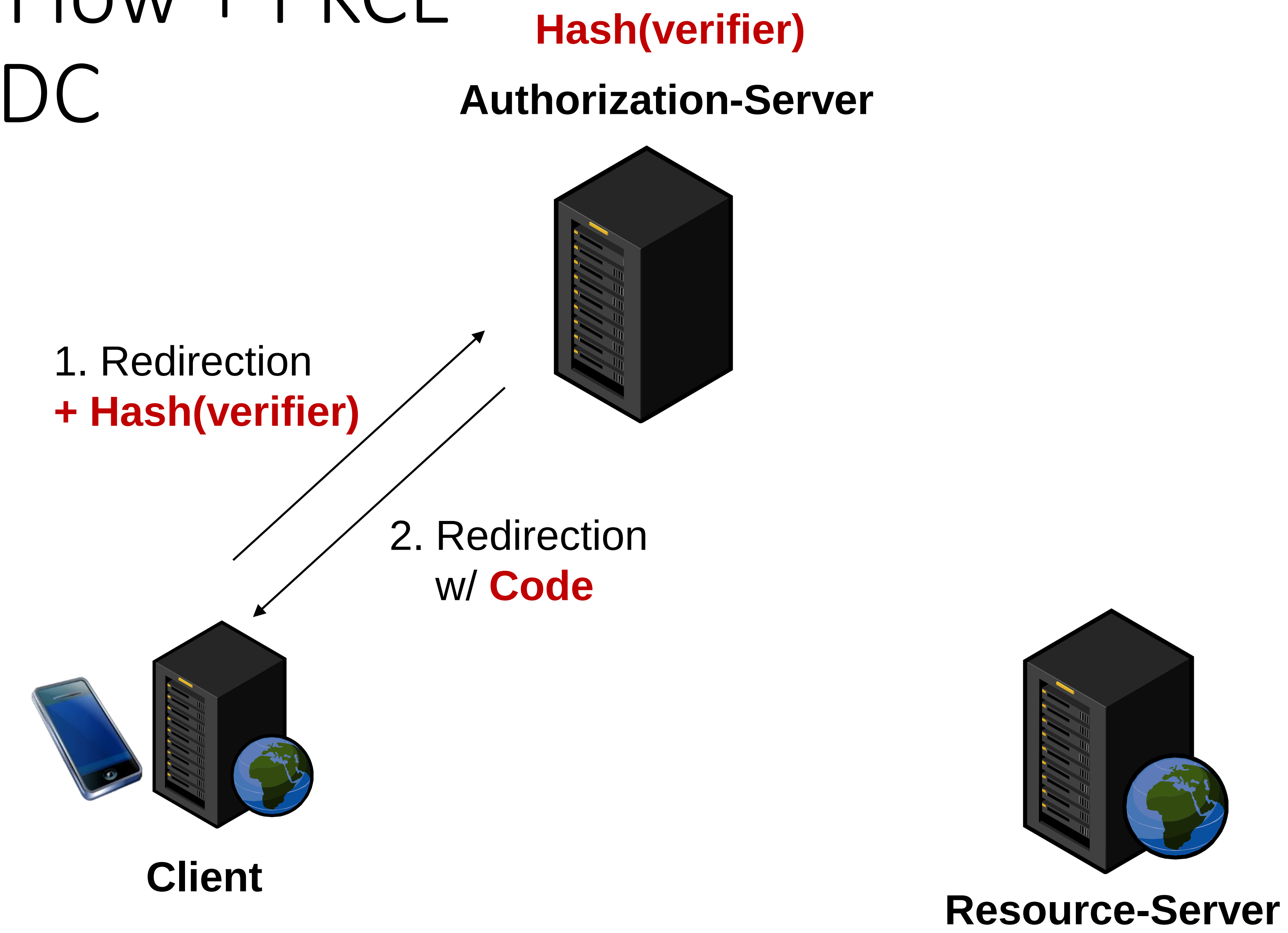
Client

Resource-Server

Code Flow w/ OIDC



Code Flow + PKCE w/ OIDC



Hash(verifier)

Authorization-Server

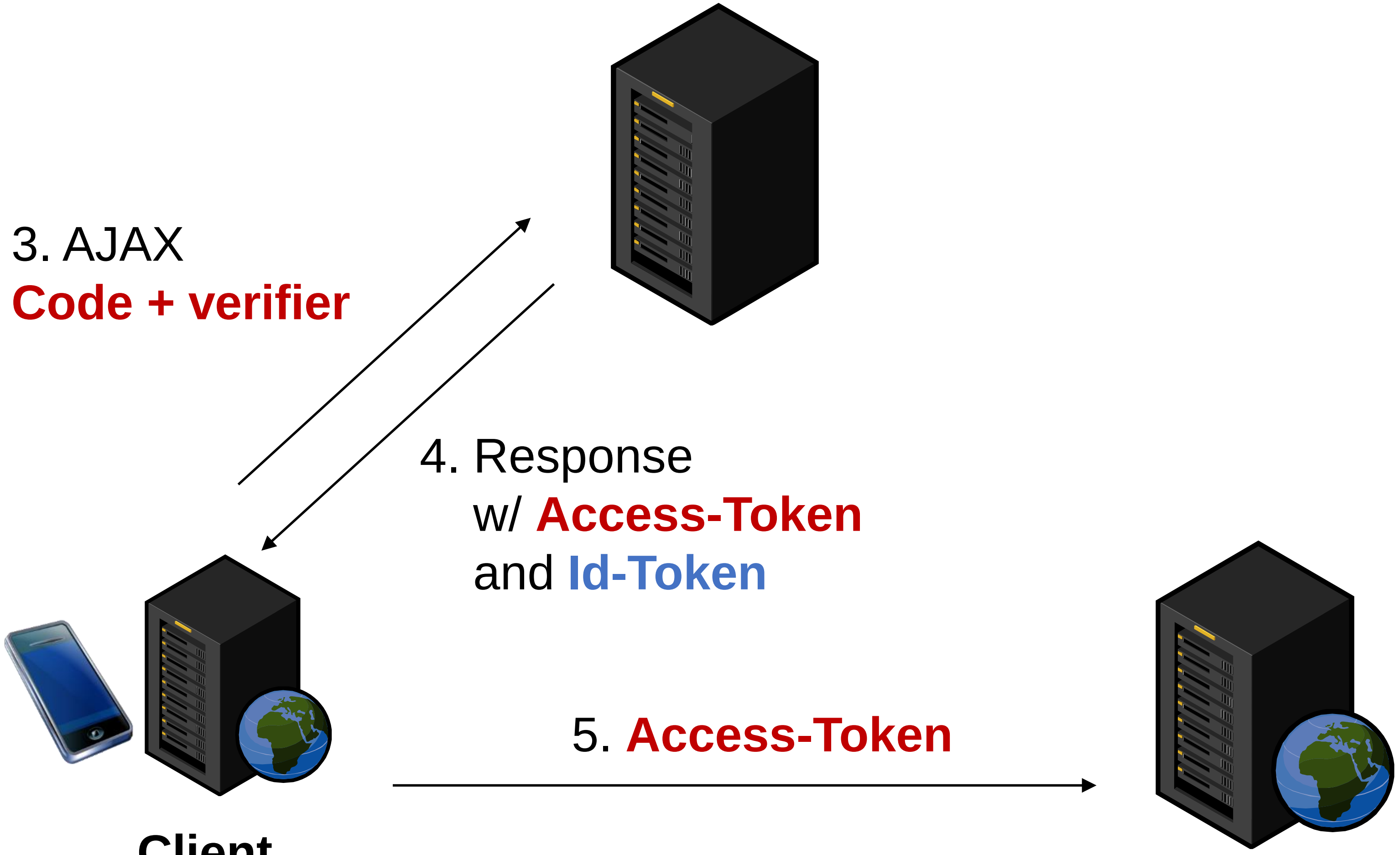
1. Redirection
+ Hash(verifier)

2. Redirection
w/ Code

Client

Resource-Server

Code Flow + PKCE w/ OIDC



Hash(verifier)

Authorization-Server

3. AJAX
Code + verifier

4. Response
w/ **Access-Token**
and **Id-Token**

5. **Access-Token**

Client

Resource-Server

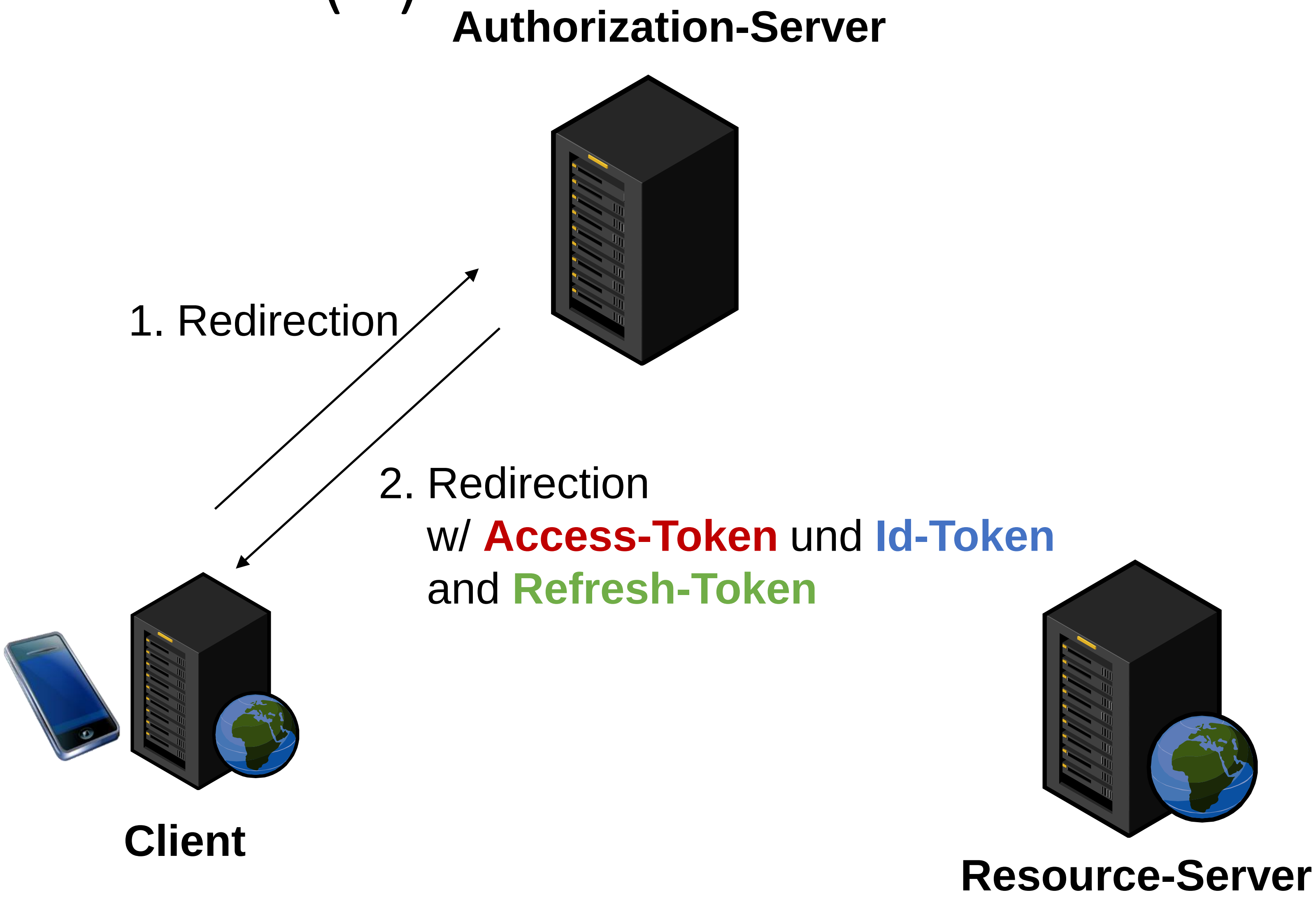
Token Refresh

Why Token Refresh?

Short living Tokens increase Security

Users don't want to login over and over again

Refresh Token (1)



Authorization-Server

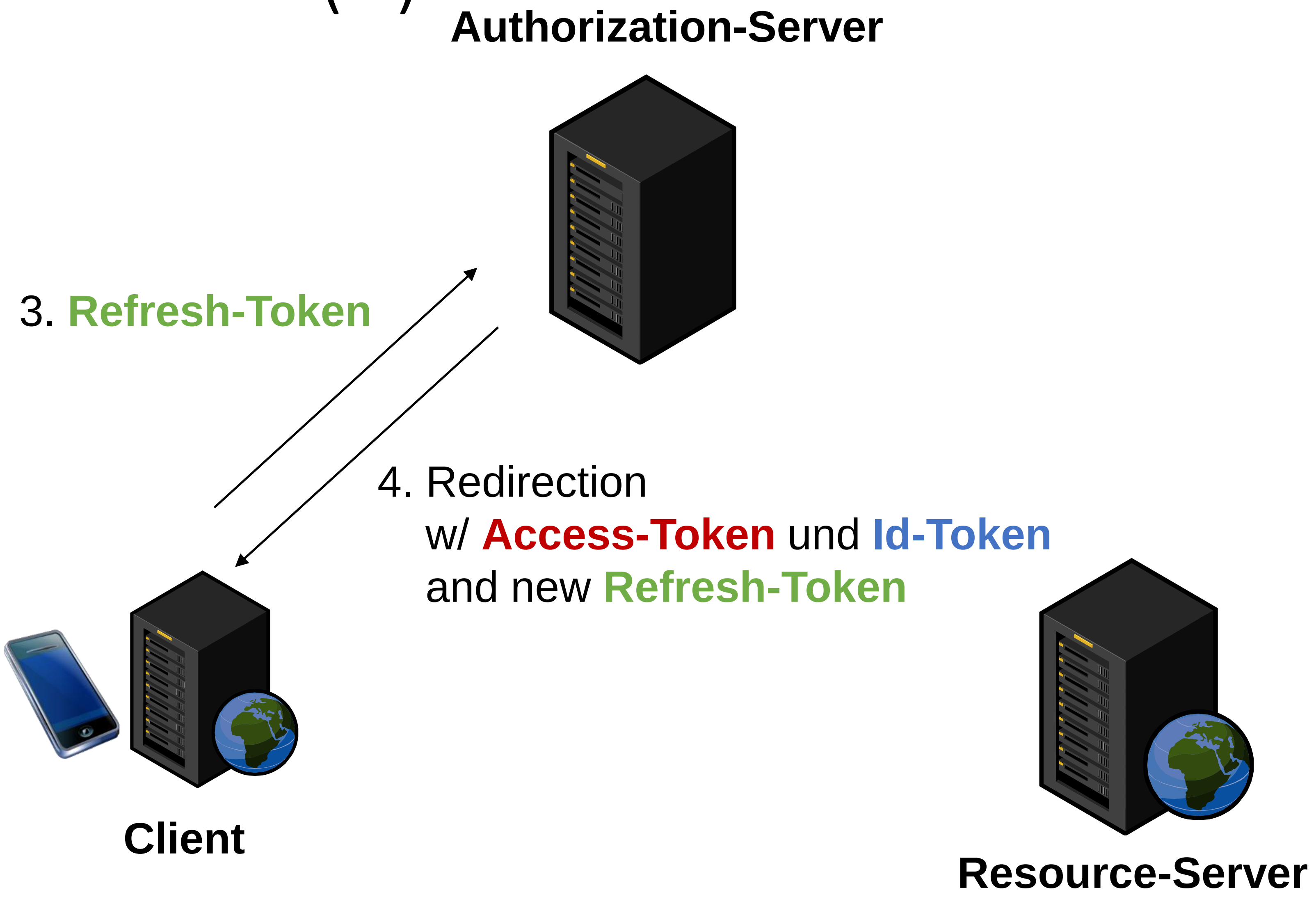
1. Redirection

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**

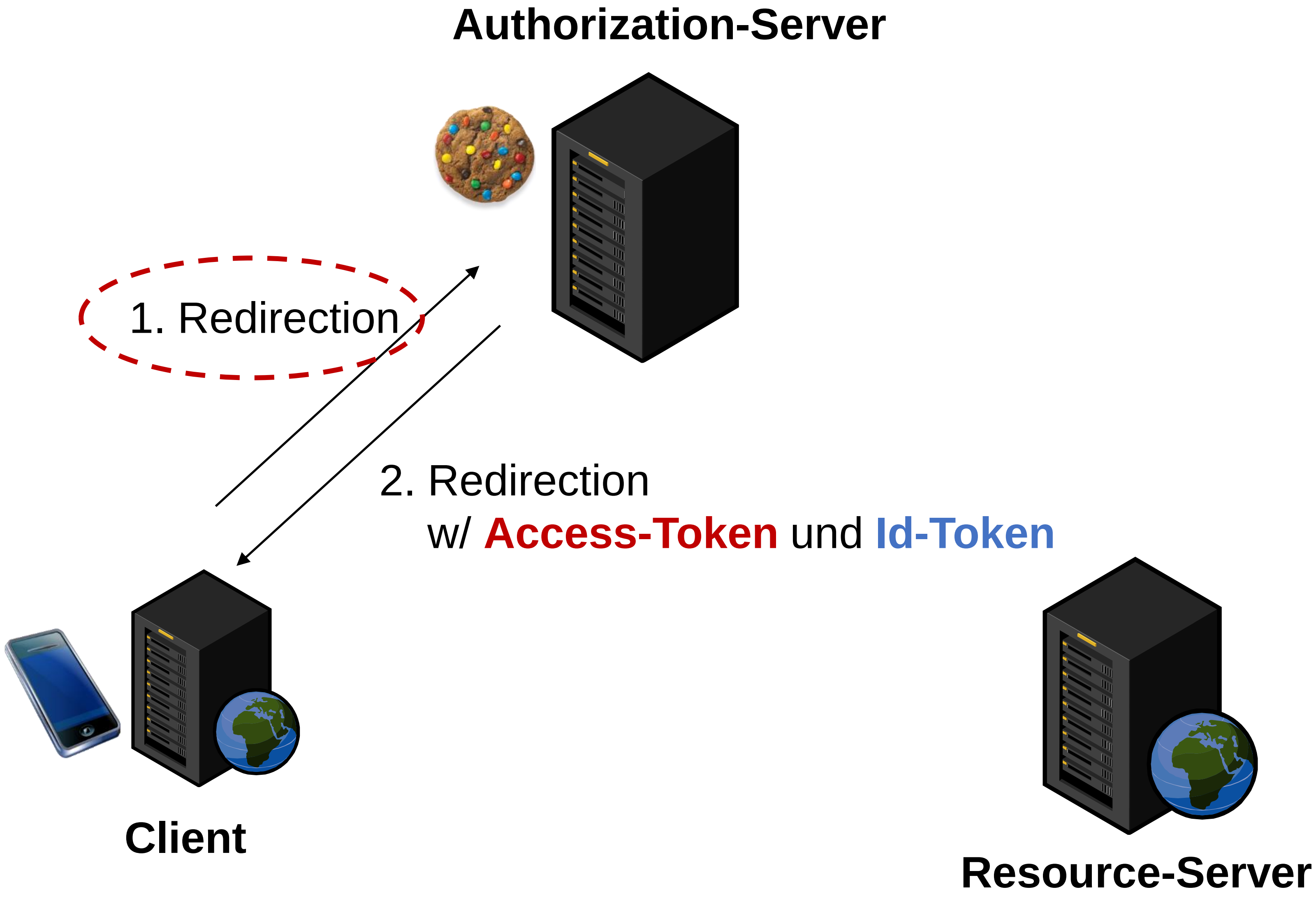
Client

Resource-Server

Refresh-Token and Browsers

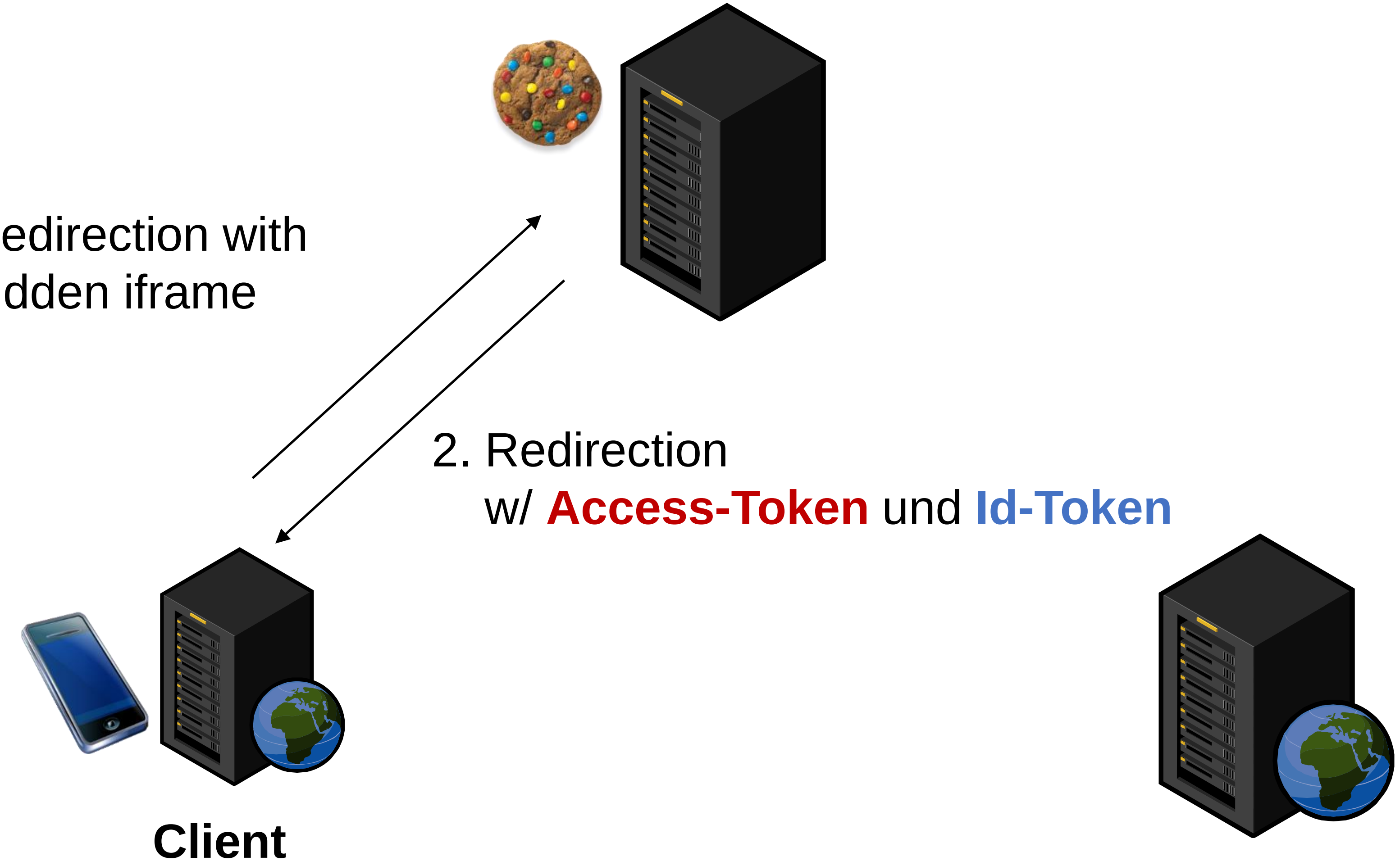
- OAuth 2.0 Security Best Current Practice allows it under specific circumstances
- Security Audit (XSS!)
- Refresh Token needs to be one-time token
- After Refresh: Client gets new refresh token
- If used by several users: log out both

Alternative: Refresh w/ Cookie



SOFTWARE architekt.at

Alternative: Silent Refresh



Authorization-Server

1. Redirection with hidden iframe

2. Redirection w/ **Access-Token** und **Id-Token**

Client

Resource-Server

DEMO

LABS

Conclusion

- Token: Flexibility, Cross Origin …
- OAuth 2: Access to Service
- OpenId Connect: SSO at Client
- Implicit Flow
- Code Flow + PKCE