



Building an Effective Architecture for Identity and Access Management

Jorge Alvarez

alvarez.jeap@gmail.com

GitHub: [@jealvarez](https://github.com/jealvarez)

Twitter: [@edlask8](https://twitter.com/edlask8)

Disclaimer

The point of views, thoughts and opinions expressed in this presentation belongs only to the presenter and not necessarily to the presenter's employer, organization, committee or other group or individual.

A dark blue, irregular ink splash or blotch serves as the background for the text. The splash has a textured, painterly appearance with some lighter blue and white areas around its edges, suggesting ink spreading on a light surface. The text "About Me" is centered within the dark blue area.

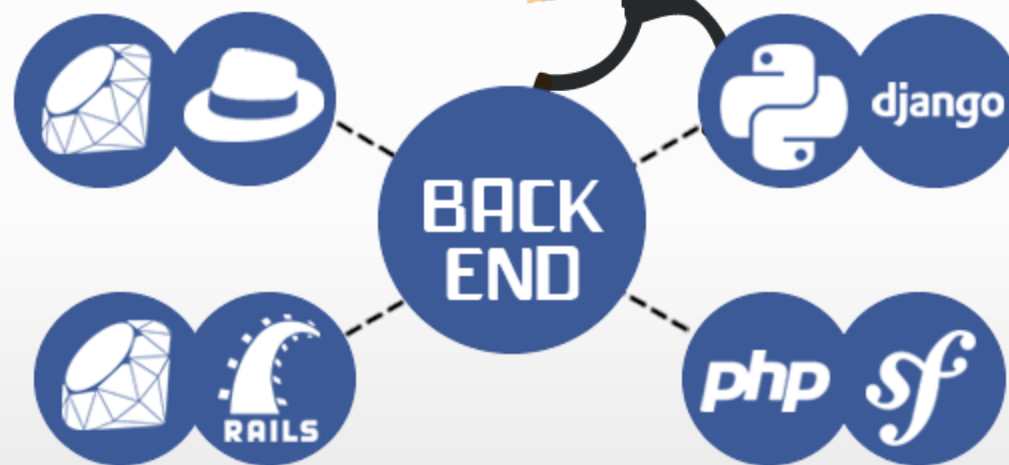
About Me

Agenda

- Challenges
- **Keycloak Overview**
- Why should I use Keycloak?
- Core Concepts
- Technology Stack
- Server Architecture
- Single Sign-on / Single Logout
- **Tokens**
- **Calling Backing Services**
- **Supported Platforms**
- Demo
- Summary
- Bibliography

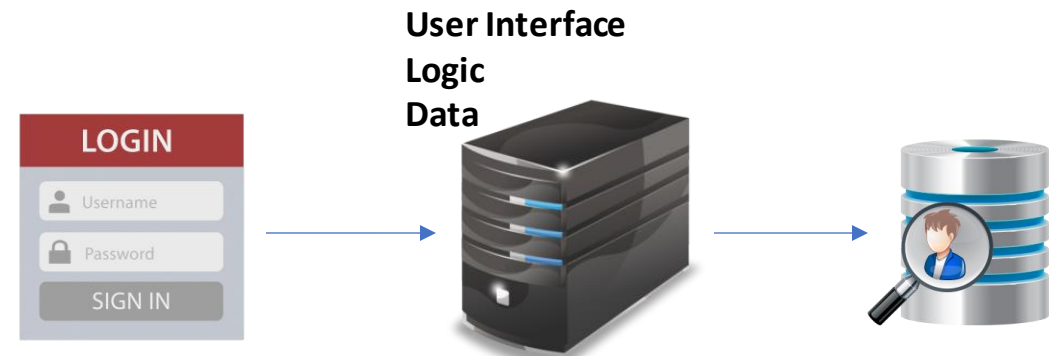
Challenges





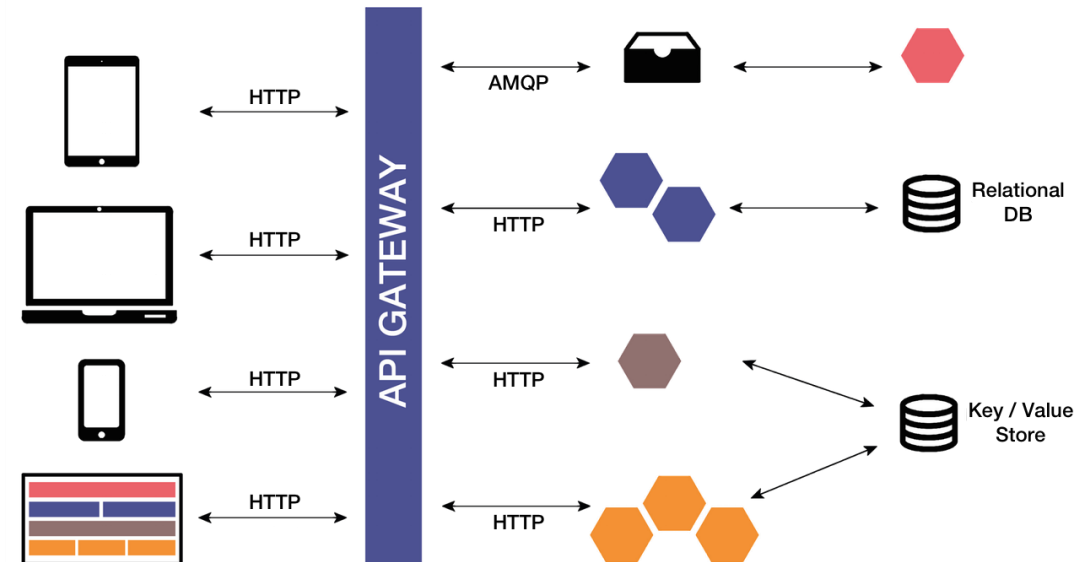
The old way

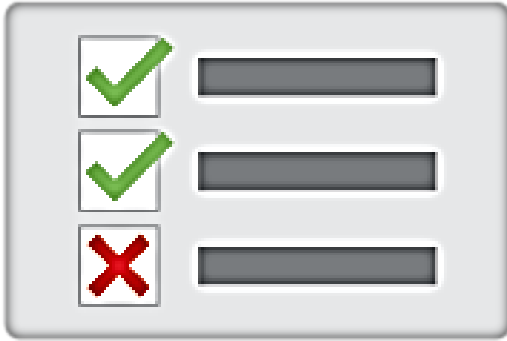
- Securing monolithic web application relatively easy
 - **Username** and **Password**
 - Credentials **verified** against **table** in **database**
 - HTTP Session **stores** in security context



The new way

- Multiple applications
- Multiple variants of each application
- Multiple services
- Multiple logins
- Multiple databases
- Multiple devices





Authorization

What you can do

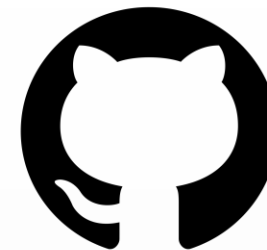
[AuthZ]



Authentication

Who you are

[AuthN]







NOW?
WHAT



Are you looking for a single sign-on solution that enables you to secure new or legacy applications and easily use federated identity providers such as social networks?



Applications
Services
Users
Devices
Permissions
Sessions
Logs
...

Do you want to manage
everything from one
console?

**You should look at
Keycloak**





Keycloak

Overview

Open Source Identity and Access Management

For Modern Applications and Services

[Get Started with Keycloak](#)

Add authentication to applications and secure services with minimum fuss. No need to deal with storing users or authenticating users. It's all available out of the box.

You'll even get advanced features such as User Federation, Identity Brokering and Social Login.

For more details go to [about](#) and [documentation](#), and don't forget to try [Keycloak](#). It's easy by design!

NEWS

05 Nov

[Keycloak 11.0.3 released](#)

04 Sep

[New Account Console](#)

31 Aug

[Keycloak 11.0.2 released](#)

Project

- Java based **AuthN** and **AuthZ** server
- Started in 2013
- Current Version **11.0.3**
 - ~ **Every 5 weeks**
- Commercial Offering Available
 - **Red Hat SSO**
 - Have you logged into **developers.redhat.com** or **www.openshift.com**?
- Community
 - **400+** Contributors
- Very **robust**, good **documentation**, many **examples**

Why should I use
Keycloak?



Features (1/3)

Adaptability

- Support multiple database engines

Integration

- Social networking logins
- Federation
 - LDAP
 - Active Directory
- Adapters for different frameworks
 - Spring
 - NodeJS
 - NetCore
 - ...

Features (2/3)

Scalability

- Clustering

Extensibility

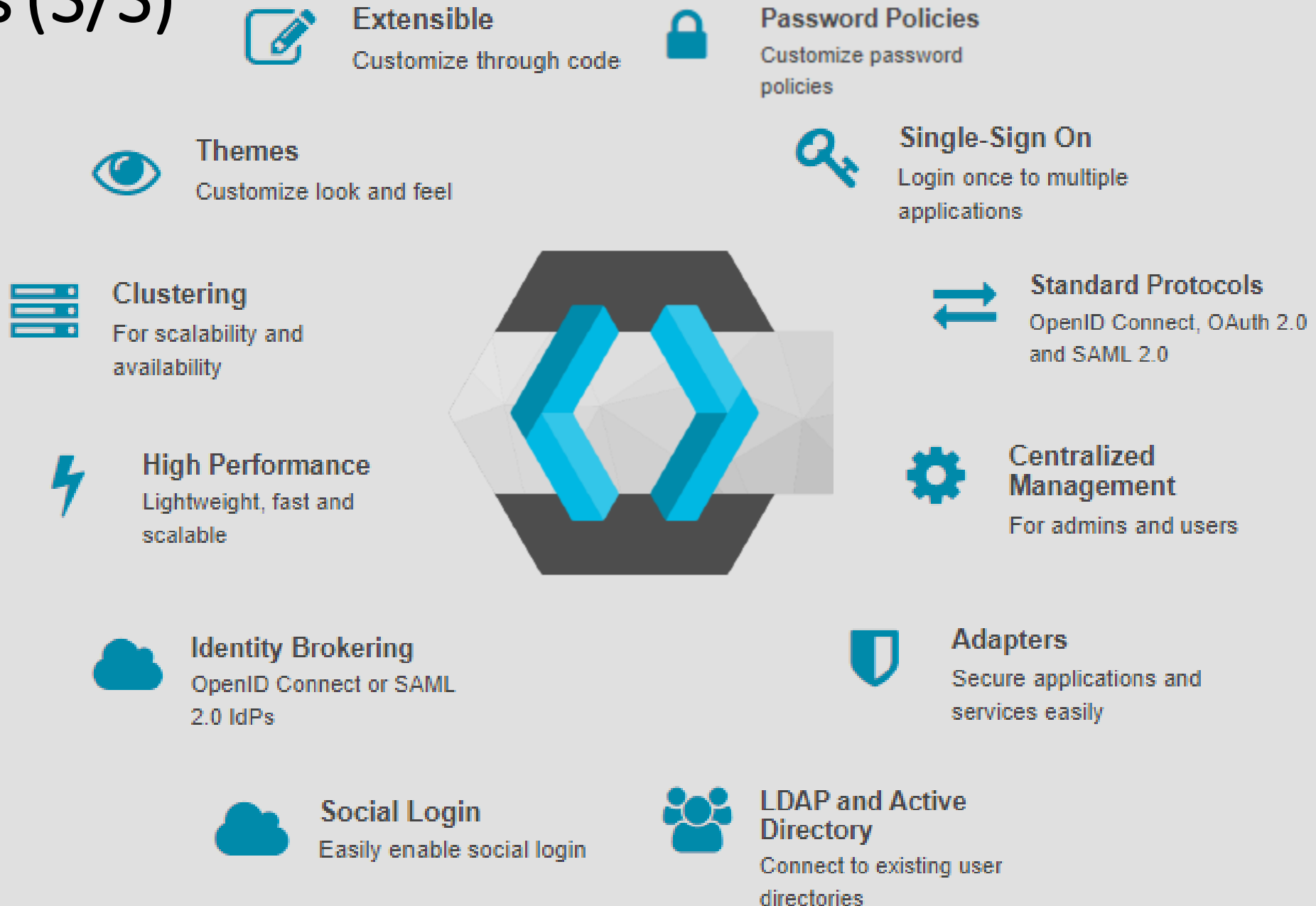
- Keycloak Service Provider Interface
 - Enables to implement your own authenticator or federator

Centralization

- Session management
 - Force logouts
 - Determine how many sessions your system currently has

Features

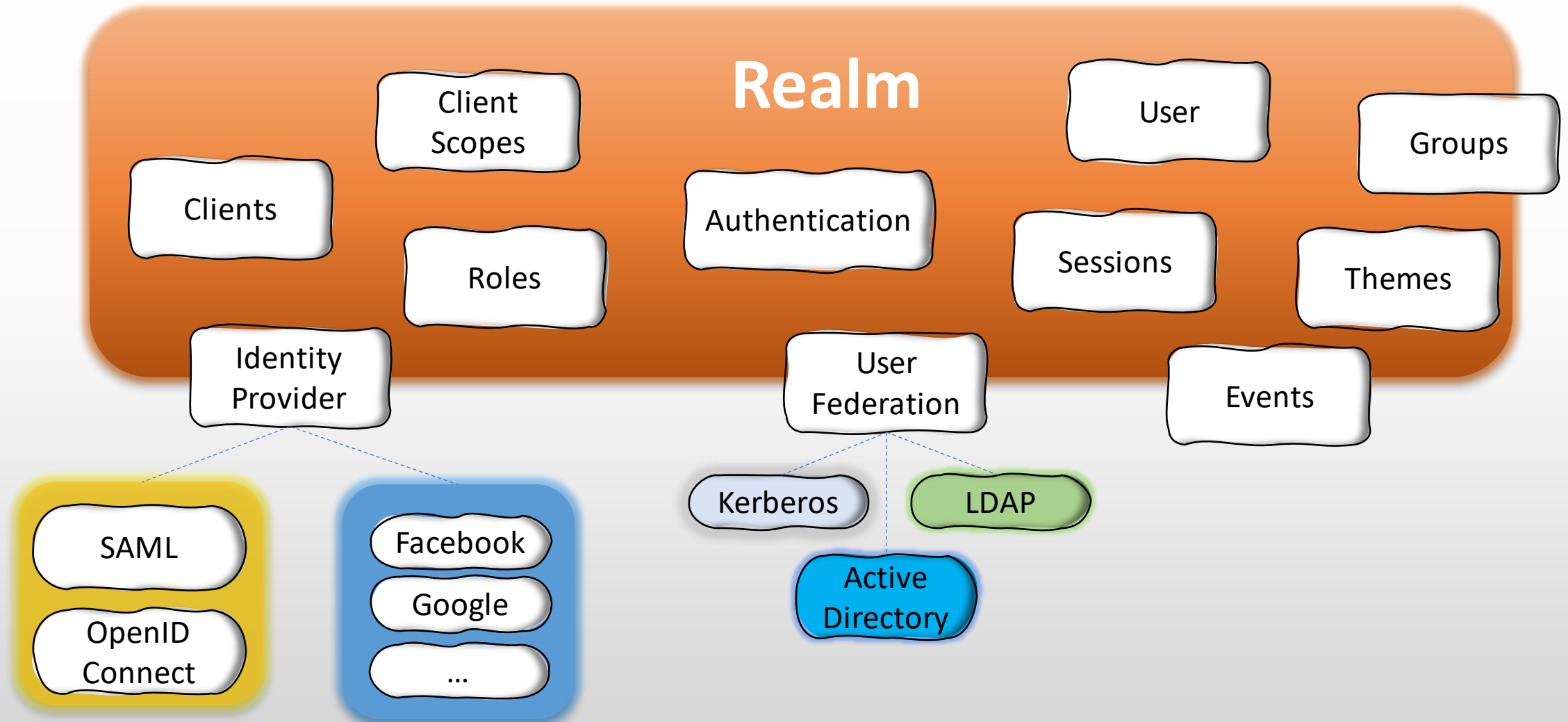
Features (3/3)



Core Concepts



Core Concepts



Technology Stack



Technology Stack

Admin Console

- ❖ AngularJS
- ❖ React
- ❖ PatternFly
- ❖ Bootstrap

Keycloak Server

- ❖ WildFly
- ❖ JPA
- ❖ RestEasy
- ❖ Freemarker
- ❖ Arquillian
- ❖ Infinispan



PATTERNFLY

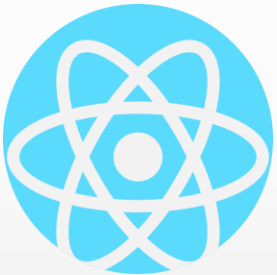
Infinispan



<#FREEMARKER>



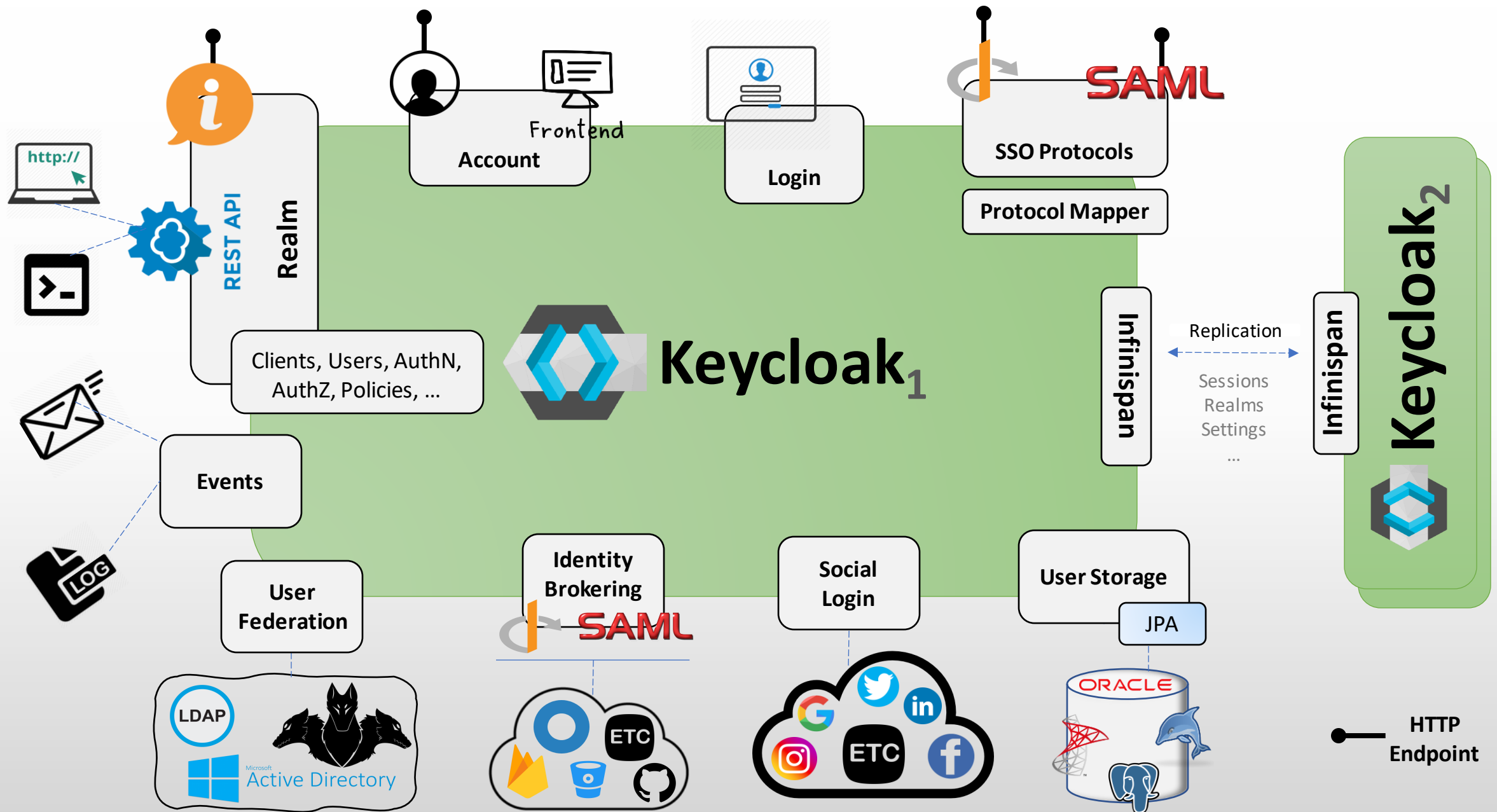
SAML



Server Architecture

Overview





Single Sign-on

How it works?



Single Sign-on

SSO

- Login **only once** to access all applications

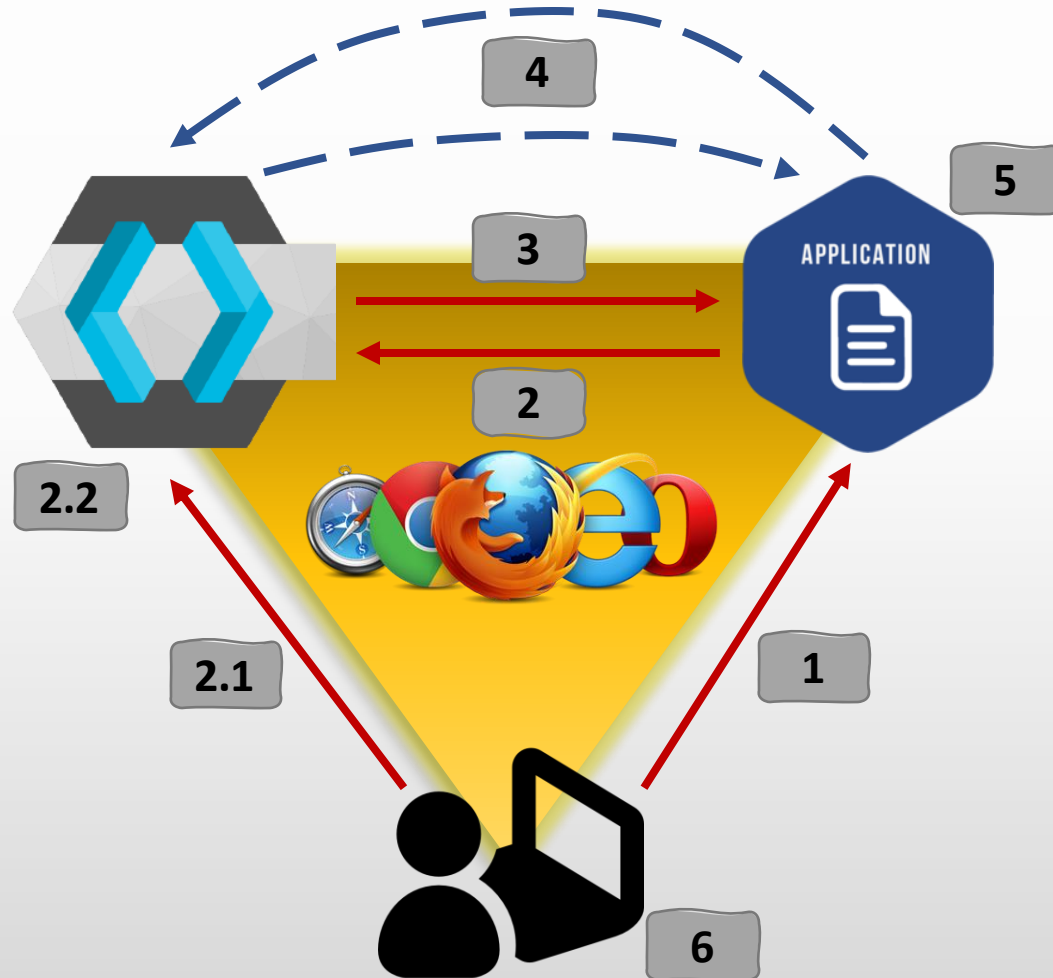
Standardized Protocols

- OpenID Connect
 - Build on top of OAuth2

Support for Single Logout

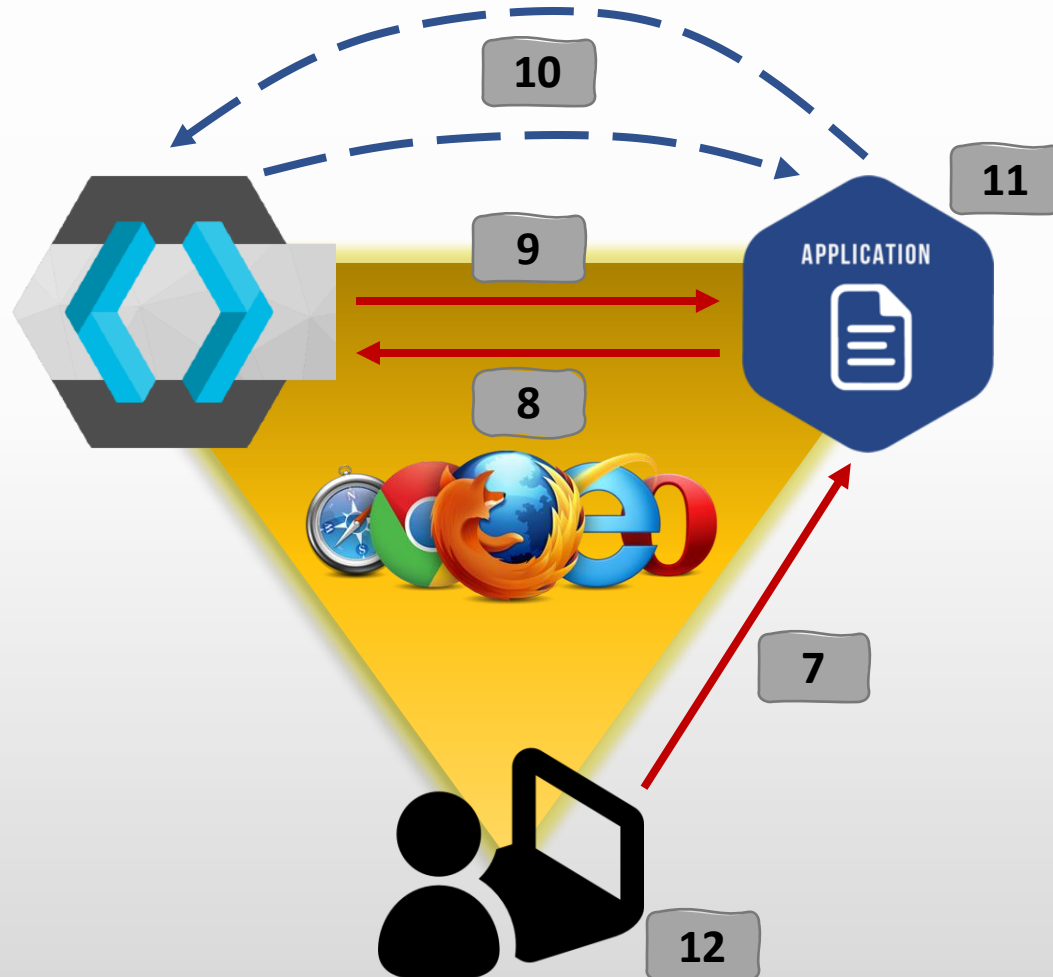
- Logouts can be propagated to applications

Unauthenticated User



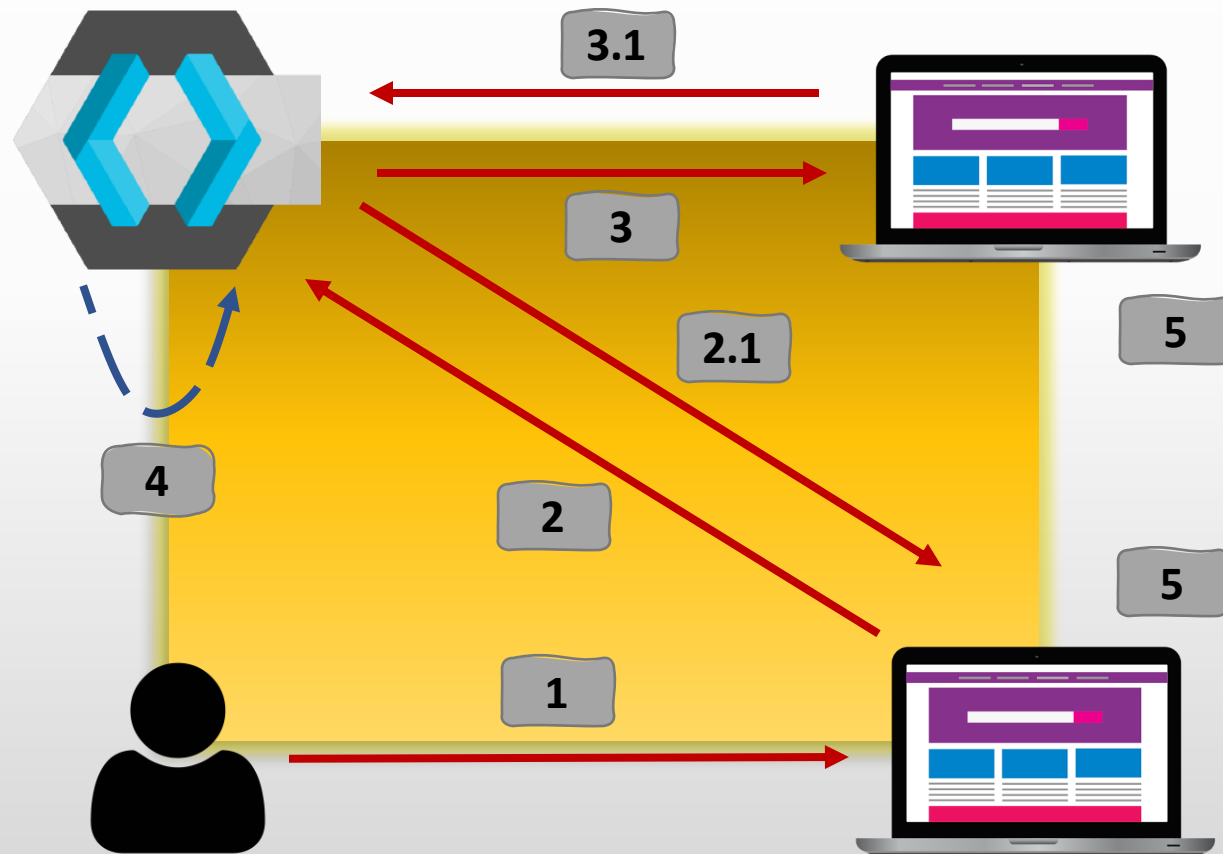
- 1 Unauthenticated user **accesses** to Application
- 2 Application **redirects** to Keycloak
 - 2.1 User **submits** credentials
 - 2.2 Credentials ✓
Keycloak **creates** SSO Session and Emits Cookies
- 3 **Generates** Code and **redirects** the user back to the Application
- 4 Application **exchanges** Code to Tokens
- 5 Application **verifies** received Tokens
Tokens are **associated** with a session
- 6 User is **signed-in** to the application

Authenticated User



- ...
- 7 Authenticated use **accesses** other application
- 8 Other application **redirects** user to Keycloak to sign-in
- 9 Keycloak **detects** SSO Session
 - Generates Code**
 - Redirects** to other application
- 10 Other application **exchanges** *code* for *tokens*
- 11 Other application **verifies** received *tokens*
 - Tokens* are **associated** with a session
- 12 User is **signed-in** to the other application

Single Logout



- 1 User **initiates** logout
- 2 Application **creates** logout request to Keycloak
 - 2.1 Keycloak **returns** response logout to the application
- 3 Keycloak **creates** logout request to **another** application
 - 3.1 Application **returns** response logout to Keycloak
- 4 Keycloak **terminates** session
- 5 Applications **do** logout

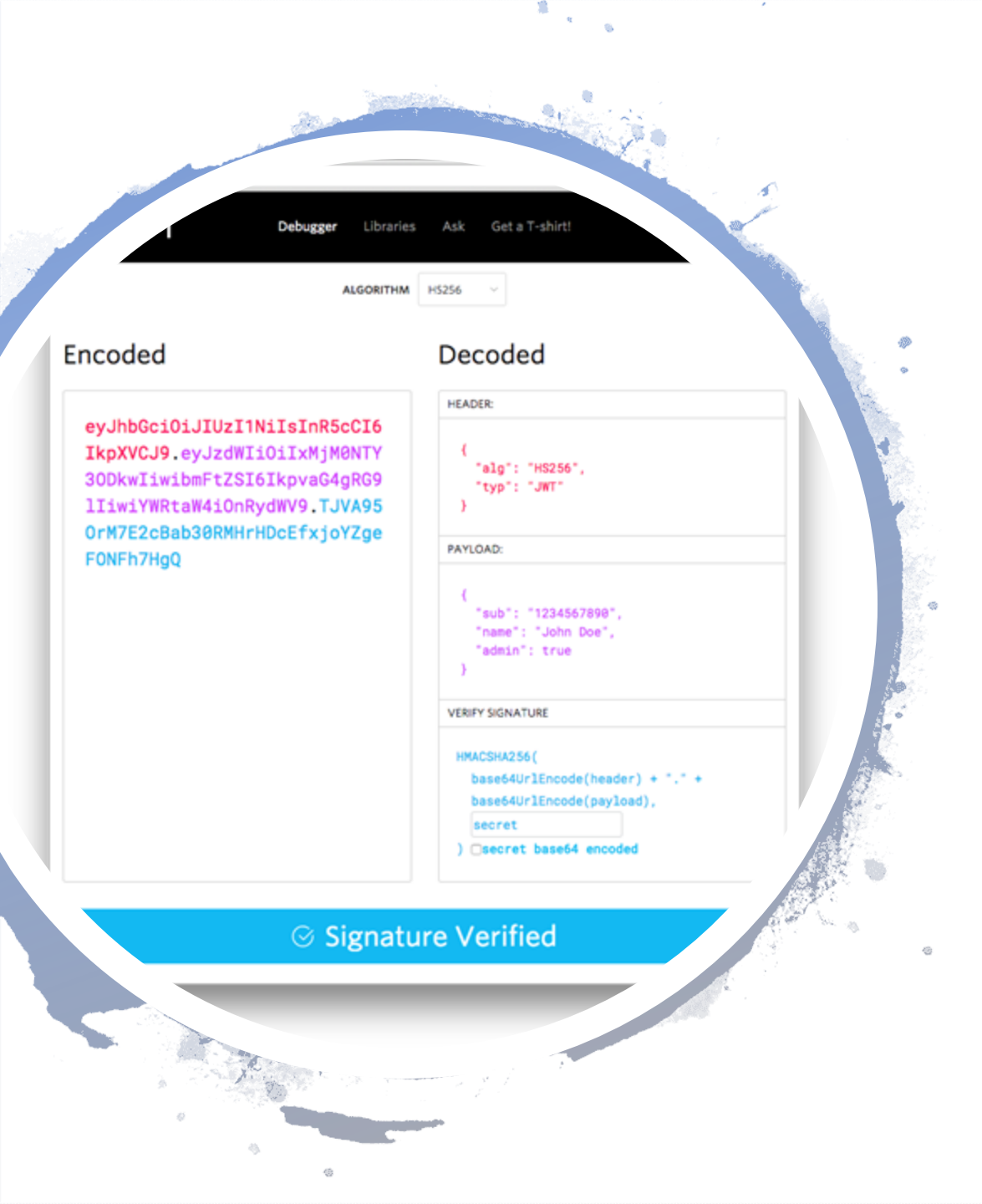
Tokens

Overview



Essential Token Types

- **Access Token**
 - Short lived [*Minutes+*]
 - Used for **accessing resources**
- **Refresh Token**
 - Long lived [*Hours+*]
 - Used for **requesting new tokens**
- **ID Token**
 - Contains **user information (OIDC)**
- **Offline Token**
 - Long lived [*Days+*]
 - **Refresh token** that **never** expires



Keycloak Tokens

- OAuth2 / OpenID Connect
 - Signed self-contained **JWT**
 - Claims
 - Key-Value Pairs + User Information + Metadata
- **Issued** by Keycloak
 - **Signed with Realm Private Key**
- **Verified** with Realm
 - Realm **Public Key**
- Limited lifespan
 - Can be revoked

JSON Web Tokens

<header-base64>. <payload-base64>. <signature-base64>



Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

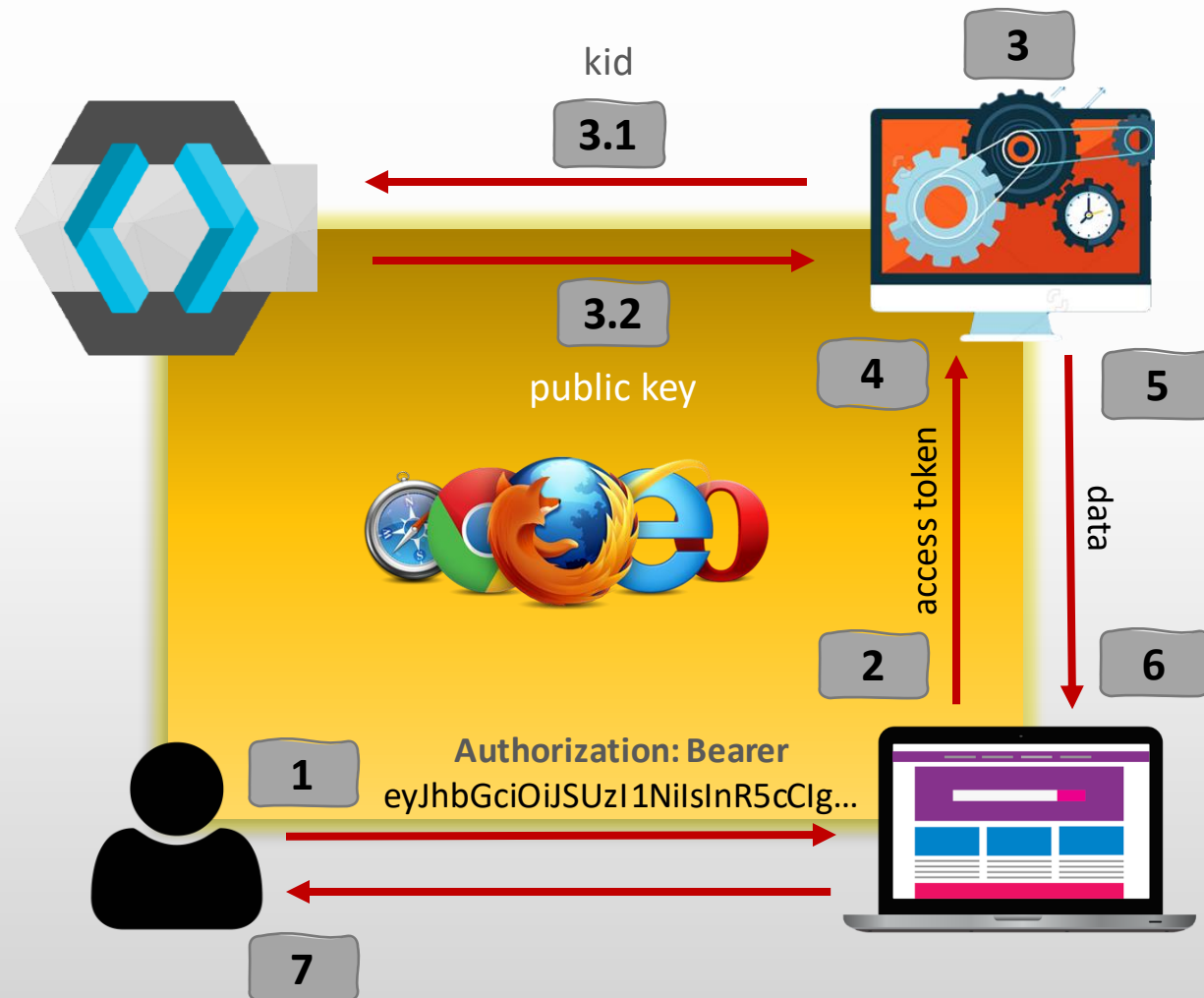
Note

Base64 means **Encoding**
Encoding != **Encryption**

Calling Backend Services



Calling Backend Services



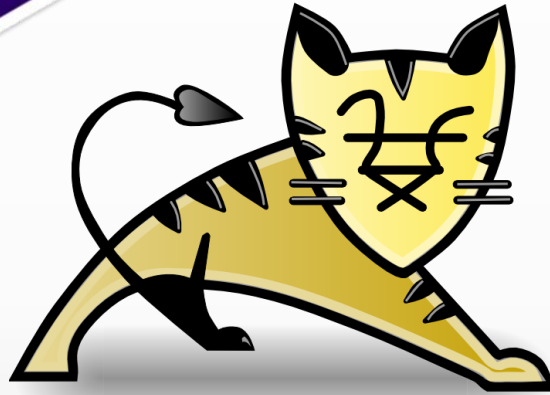
- 1 Authenticated user **accesses** to the application
- 2 Application **uses** the **access token** in the **http header** to **access** to the backend
- 3 The backend **looks up** the **Realm Public Key** in cache with the **kid** from the **JWT**
 - 3.1 If **not found**, then **fetch** Public Key with **kid** from Keycloak
 - 3.2 Keycloak **returns** Realm Public Key
- 4 The backend **verifies** signature of the **access token** with the Realm Public Key
- 5 The backend **grants** access and **returns** data
- 6 The application **can** display the data
- 7 User can **access** to the data

Supported Platforms





SAML



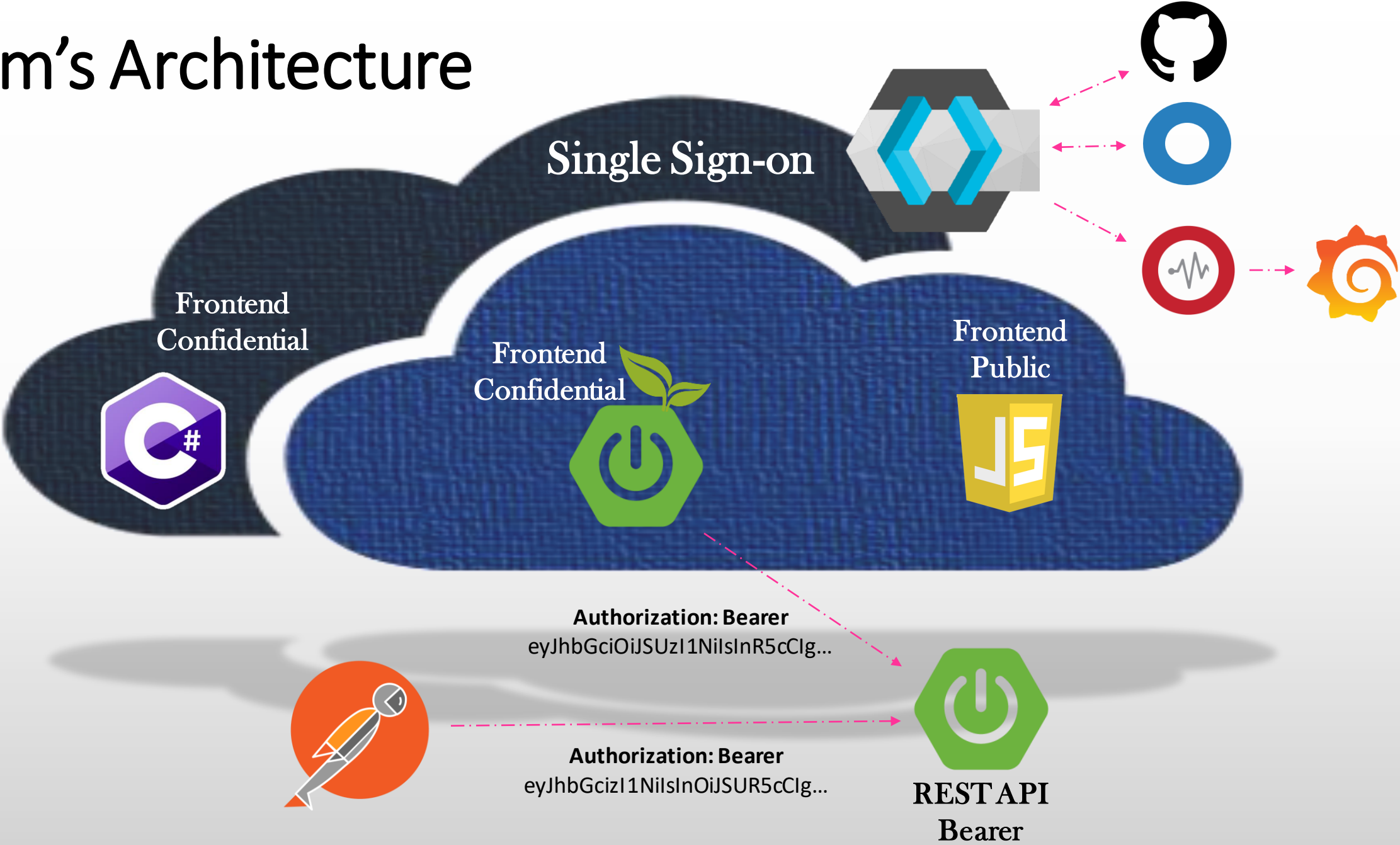
APACHE
HTTP SERVER

I DON'T KNOW RICK

LOOKS FAKE TO ME



System's Architecture



Summary



Summary (1/2)

So easy to get started with

- Unzip and Run
- Docker Images

Provides many features out of the box

- Single Sign-on
- Single Logout
- Federation
- User Management
- Social Logins
- ...

Summary (2/2)

Build on proven a robust standards

- OAuth 2.0
- OpenID Connect 1.0
- SAML 2.0

Extensible

- Custom
 - Authentication mechanisms
 - Event Listeners
 - Themes
 - ...

Easy to integrate

- Adapters available for different frameworks

...

Bibliography



Useful links

[Keycloak
Website](#)

[Keycloak
Community
Extensions](#)

[Keycloak
Docker Images](#)

[Keycloak
Quickstart
Projects](#)

[OpenID
Connect](#)

[SAML](#)

[JSON Web
Tokens](#)



Email: alvarez.jeap@gmail.com

Github: [@jealvarez](https://github.com/jealvarez)

Twitter: [@edlask8](https://twitter.com/edlask8)