Building an Effective Architecture for Identity and Access Management

Jorge Alvarez
alvarez.jeap@gmail.com
GitHub: @jealvarez
Twitter: @edlask8

Disclaimer

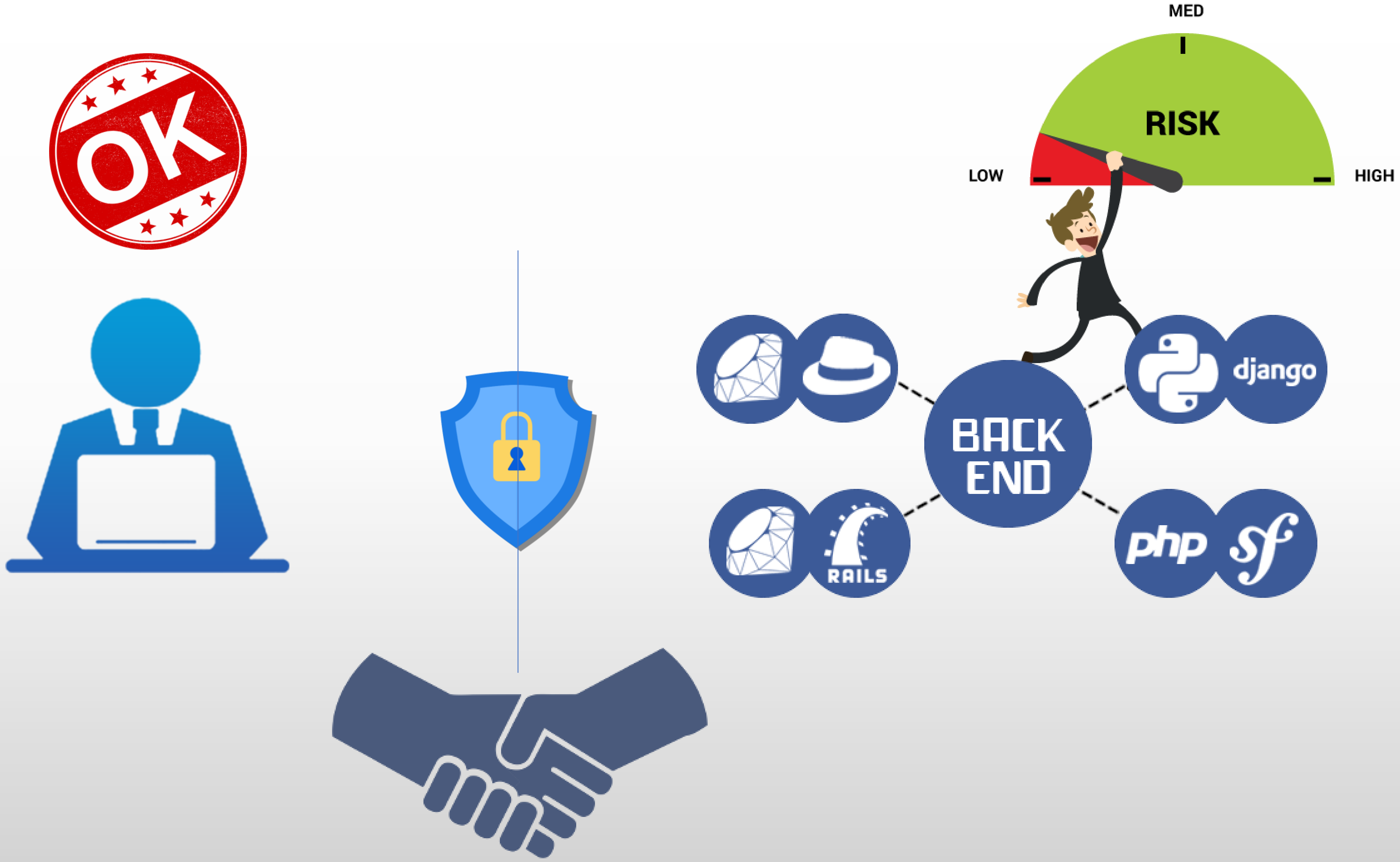
The point of views, thoughts and opinions expressed in this presentation belongs only to the presenter and not necessarily to the presenter's employer, organization, committee or other group or individual.

About Me

Agenda

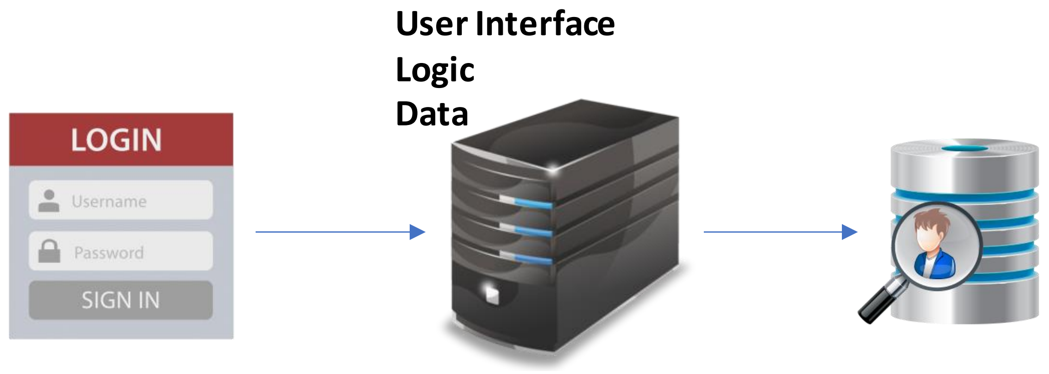
- **Challenges**
- **Keycloak Overview**
- Why should I use Keycloak?
- Core Concepts
- Technology Stack
- Server Architecture
- Single Sign-on / Single Logout
- **Tokens**
- **Calling Backing Services**
- **Supported Platforms**
- **Demo**
- **Summary**
- **Bibliography**

Challenges


OK
MED
RISK
LOW
HIGH
BACK END
django
php
RAILS

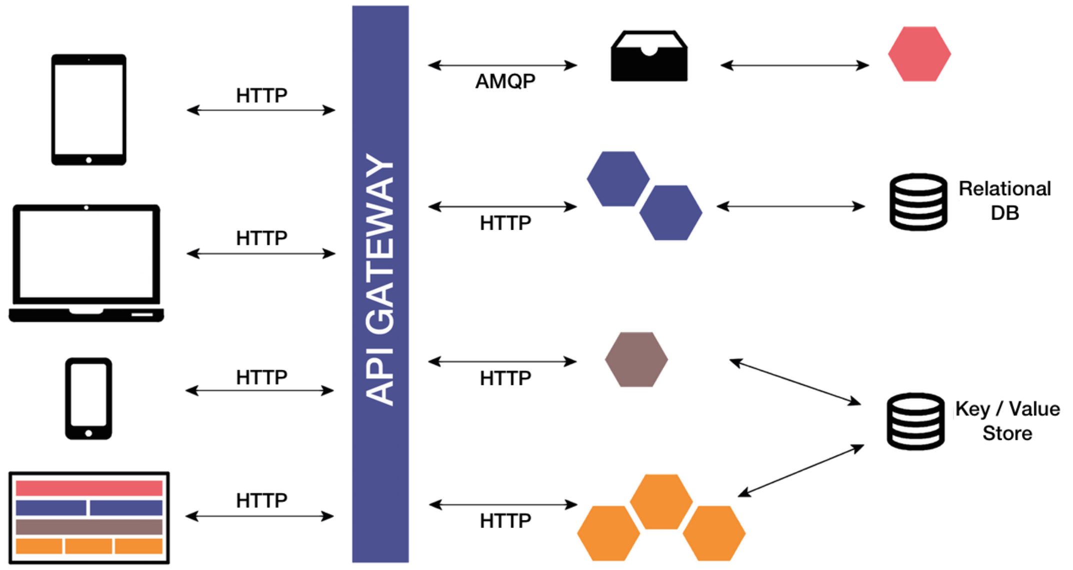
The old way

- 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]

LDAP

in

Microsoft
Active Directory

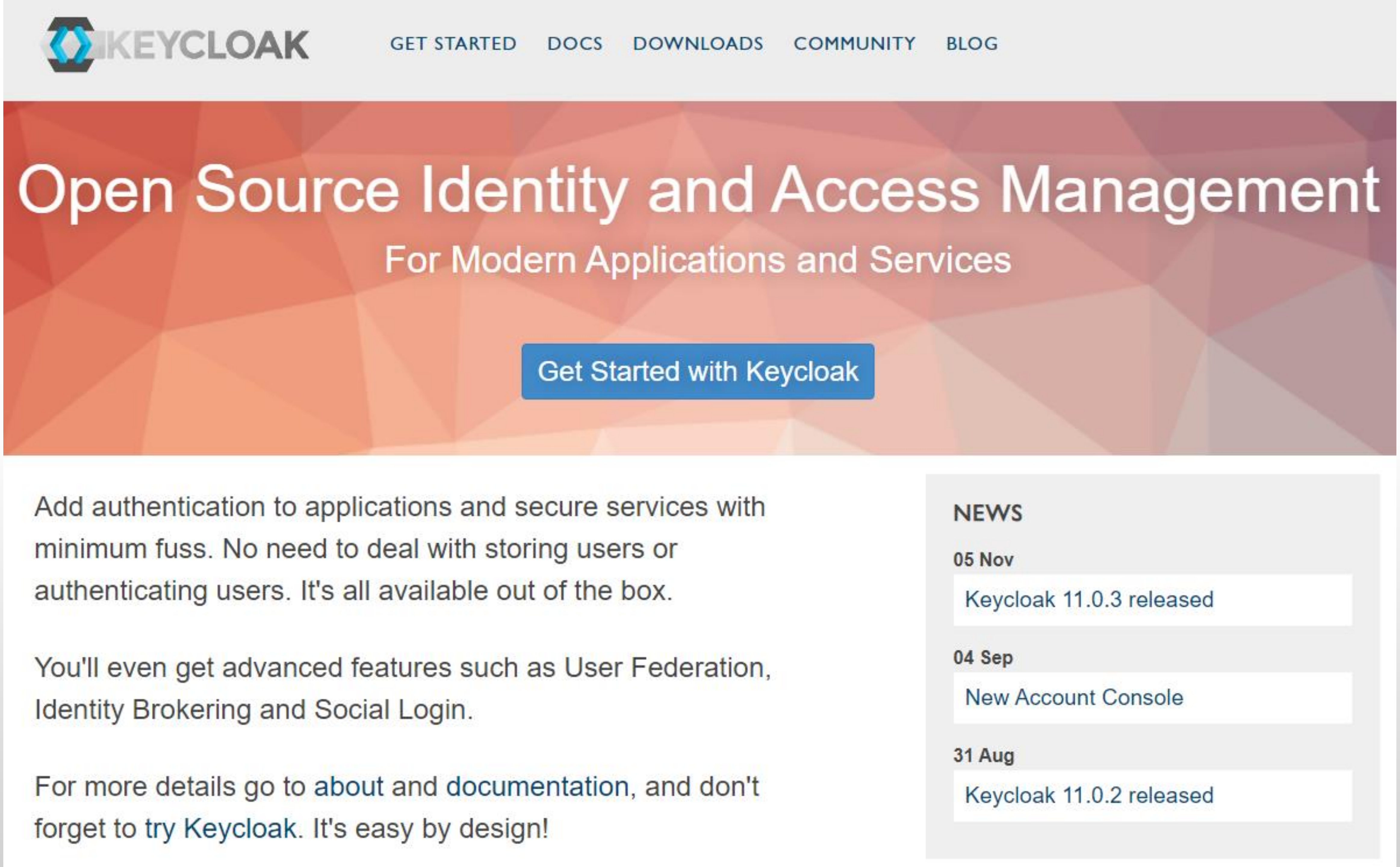
NOW WHAT?

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?

You should look at Keycloak

Keycloak

Overview



Open Source Identity and Access Management

For Modern Applications and Services

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)

Extensible
Customize through code

Password Policies
Customize password policies

Themes
Customize look and feel

Single-Sign On
Login once to multiple applications

Clustering
For scalability and availability

Standard Protocols
OpenID Connect, OAuth 2.0 and SAML 2.0

High Performance
Lightweight, fast and scalable

Centralized Management
For admins and users

Identity Brokering
OpenID Connect or SAML 2.0 IdPs

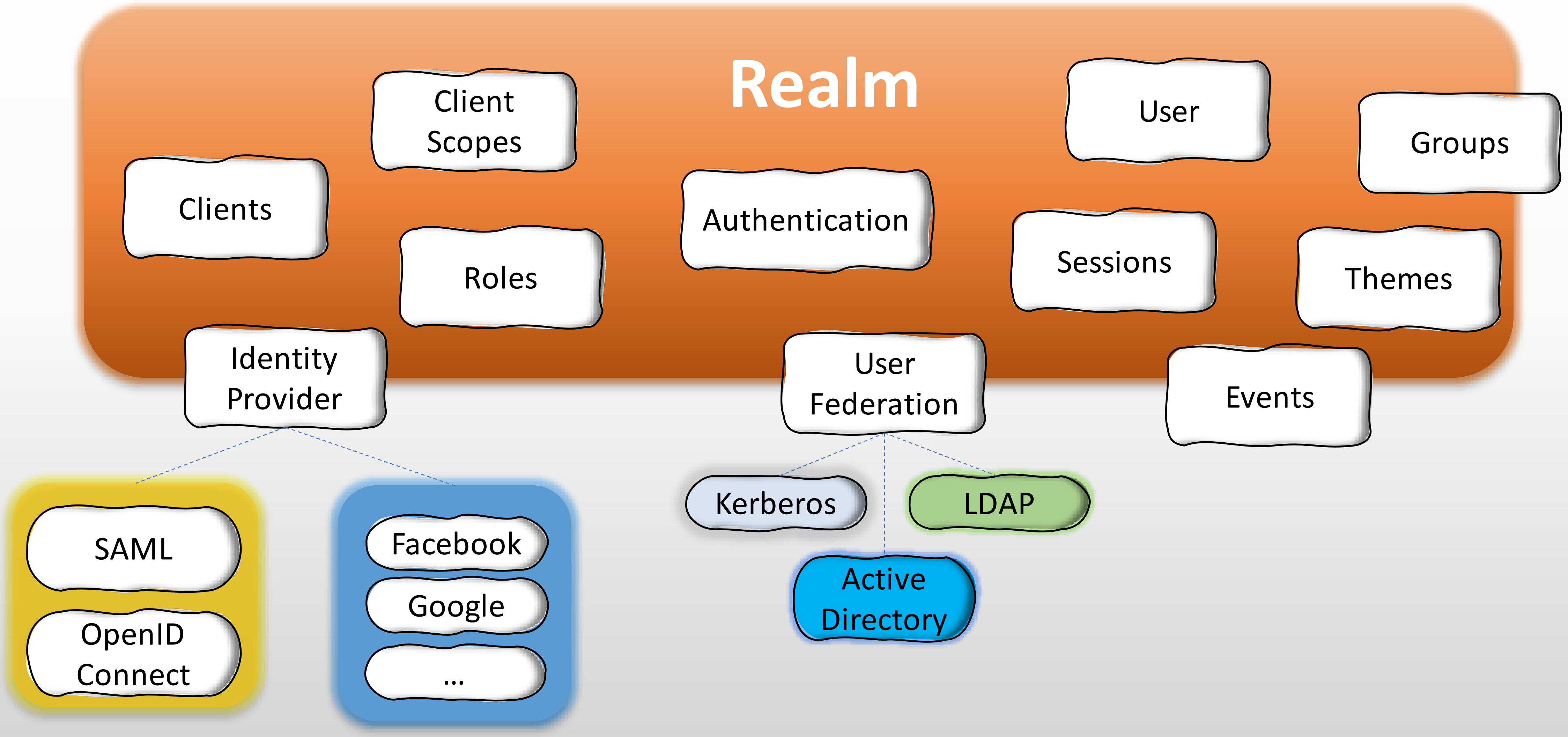
Adapters
Secure applications and services easily

Social Login
Easily enable social login

LDAP and Active Directory
Connect to existing user directories

Core Concepts

Core Concepts



Technology Stack

Technology Stack

Admin Console

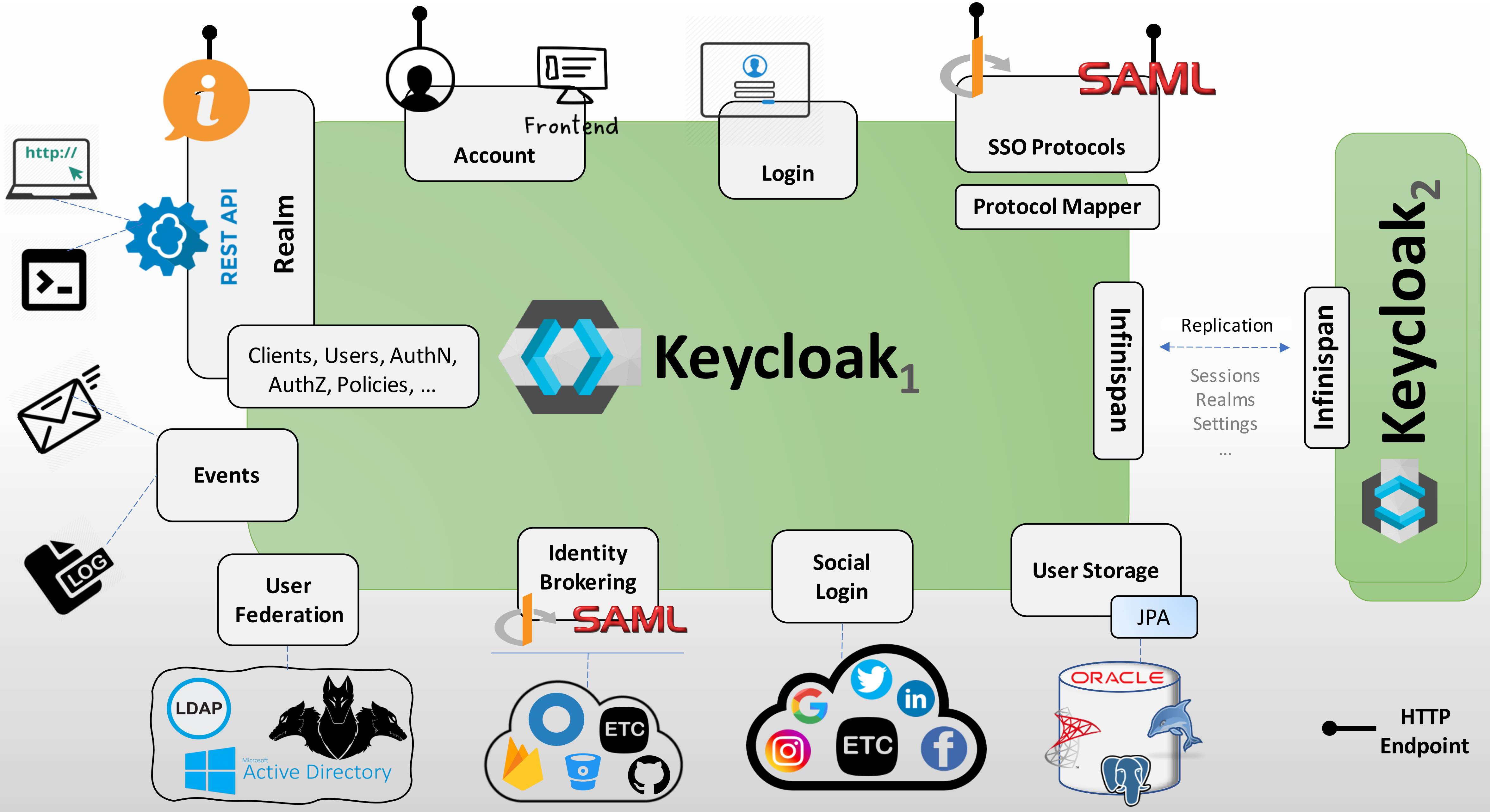
- AngularJS
- React
- PatternFly
- Bootstrap

Keycloak Server

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

Server Architecture

Overview


http://
REST API
Realm
Account
Frontend
Login
SSO Protocols
SAML
Protocol Mapper
Clients, Users, AuthN, AuthZ, Policies, ...
Keycloak₁
Infinispan
Replication
Sessions
Realms
Settings
...
Infinispan
Keycloak₂
Events
LOG
User Federation
LDAP
Microsoft Active Directory
Identity Brokering
SAML
ETC
Social Login
ETC
User Storage
JPA
ORACLE
HTTP Endpoint

Single Sign-on

How it works?

Single Sign-on

SSO

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

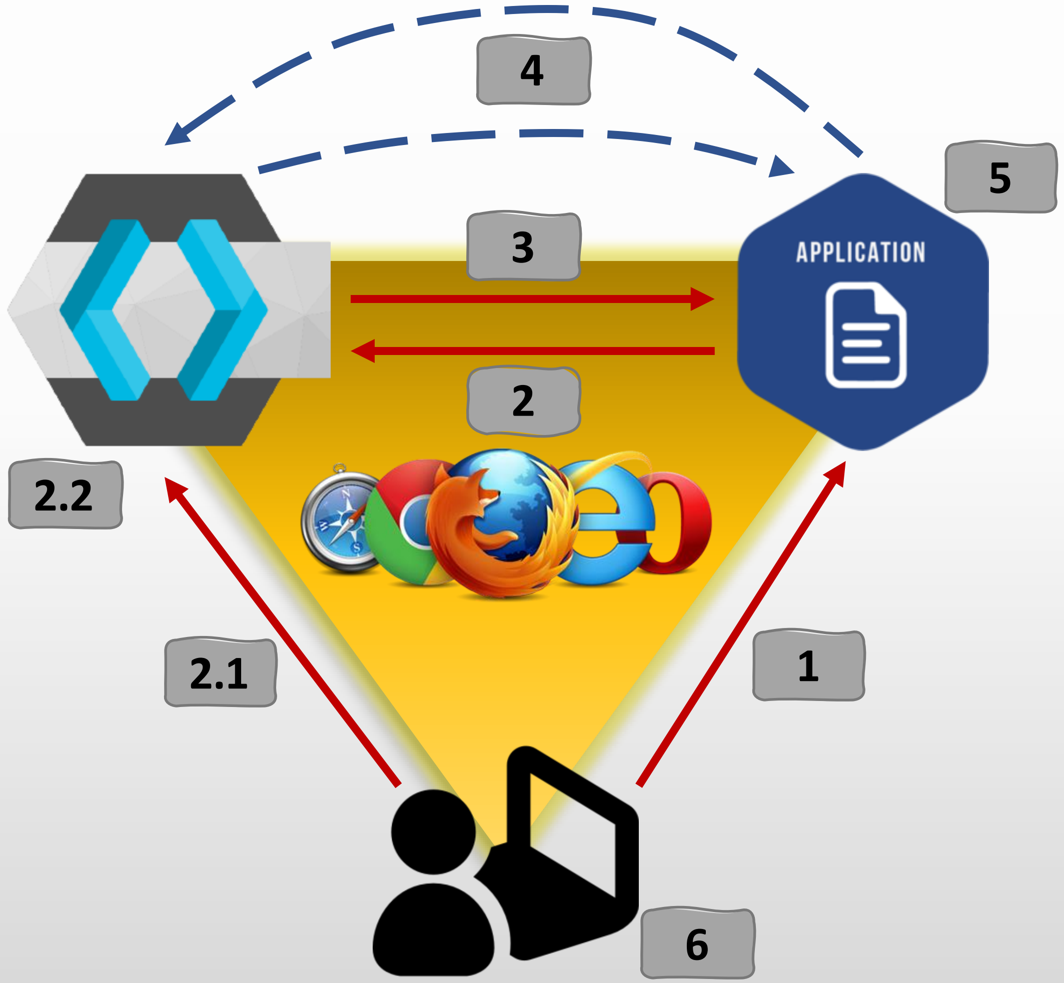
Standardized Protocols

- OpenID Connect
  - Build on top of OAuth2

Support for **Single Logout**

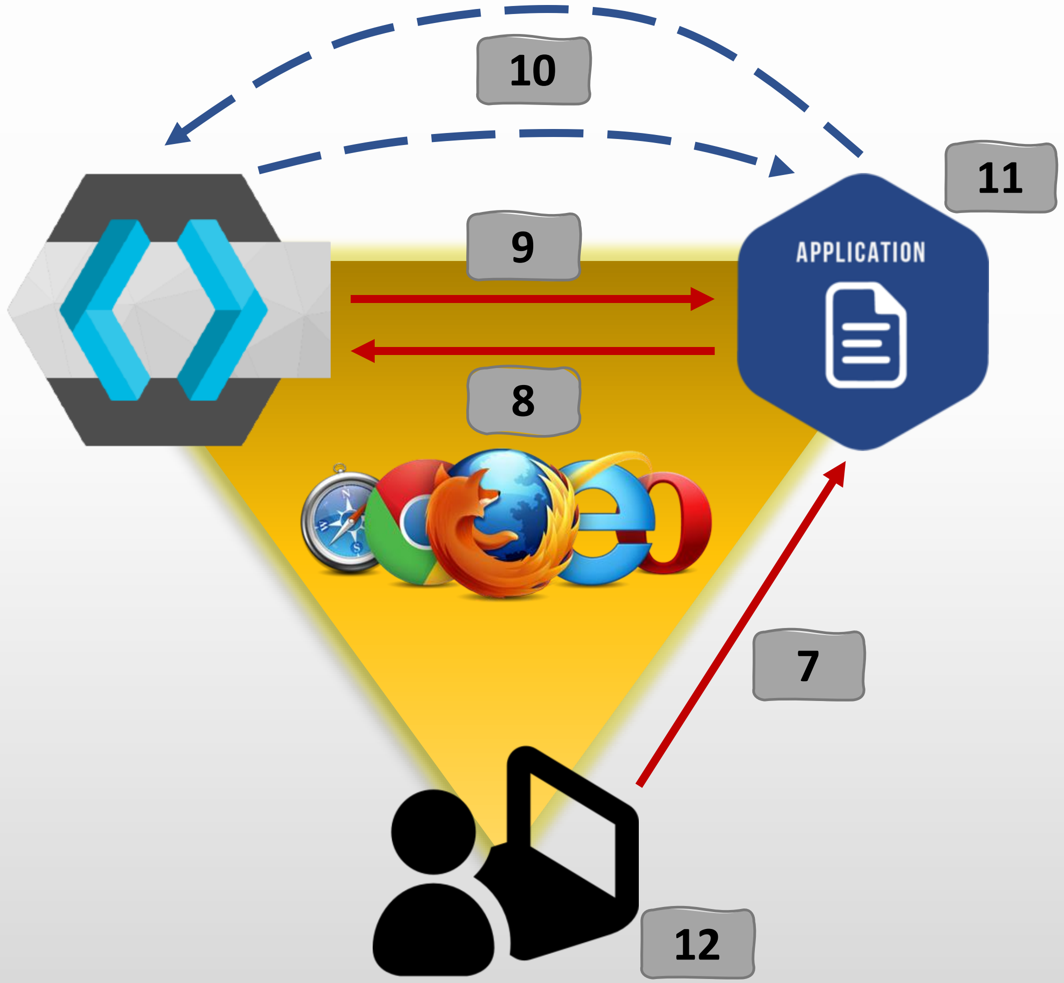
- Logouts can be propagated to applications

Unauthenticated User



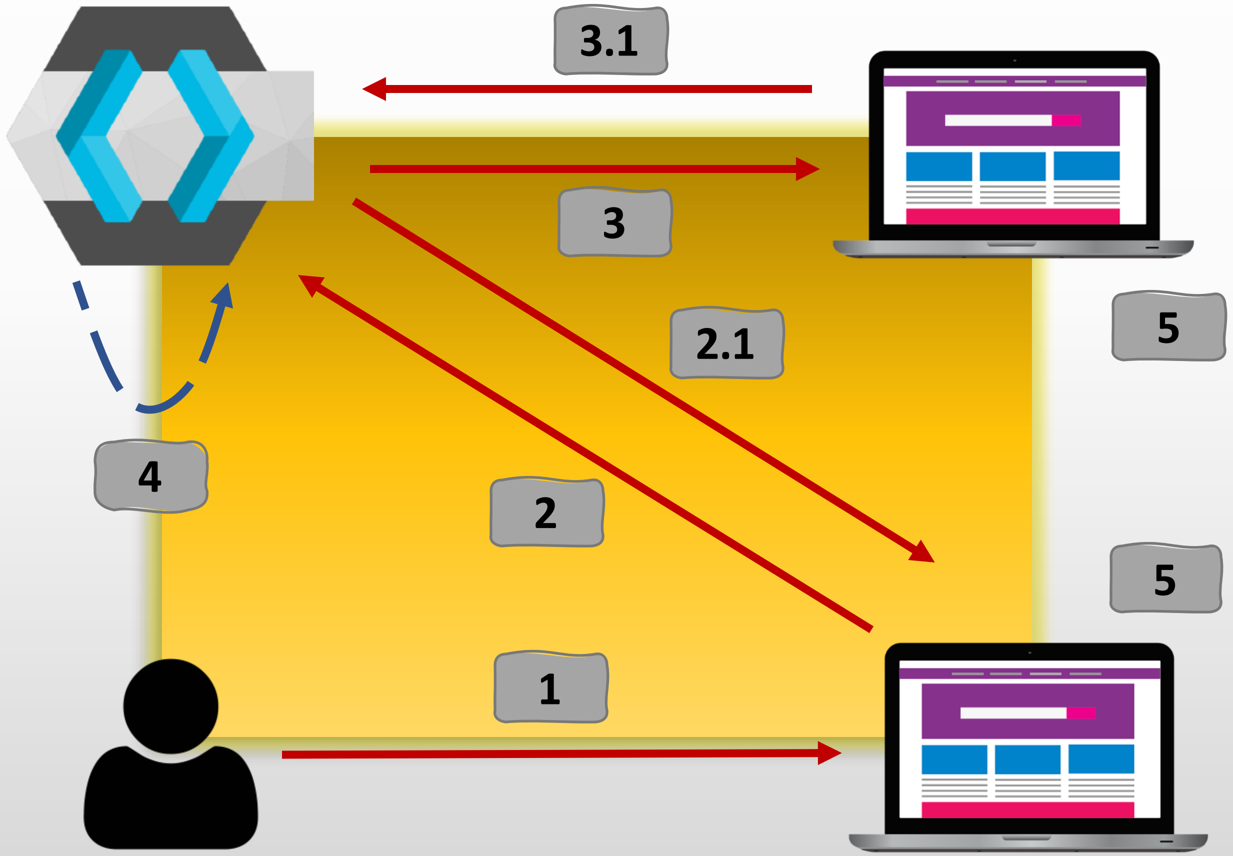
1	Unauthenticated user **accesses** to Application
2	Application **redirects** to Keycloak
2.1	User **submits** credentials
2.2	Credentials ✓ Keycloak **creates** SSO Session and Emits Cookies
3	**Generates** *Code* and **redirects** the user back to the Application
4	Application **exchanges** *Code* 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



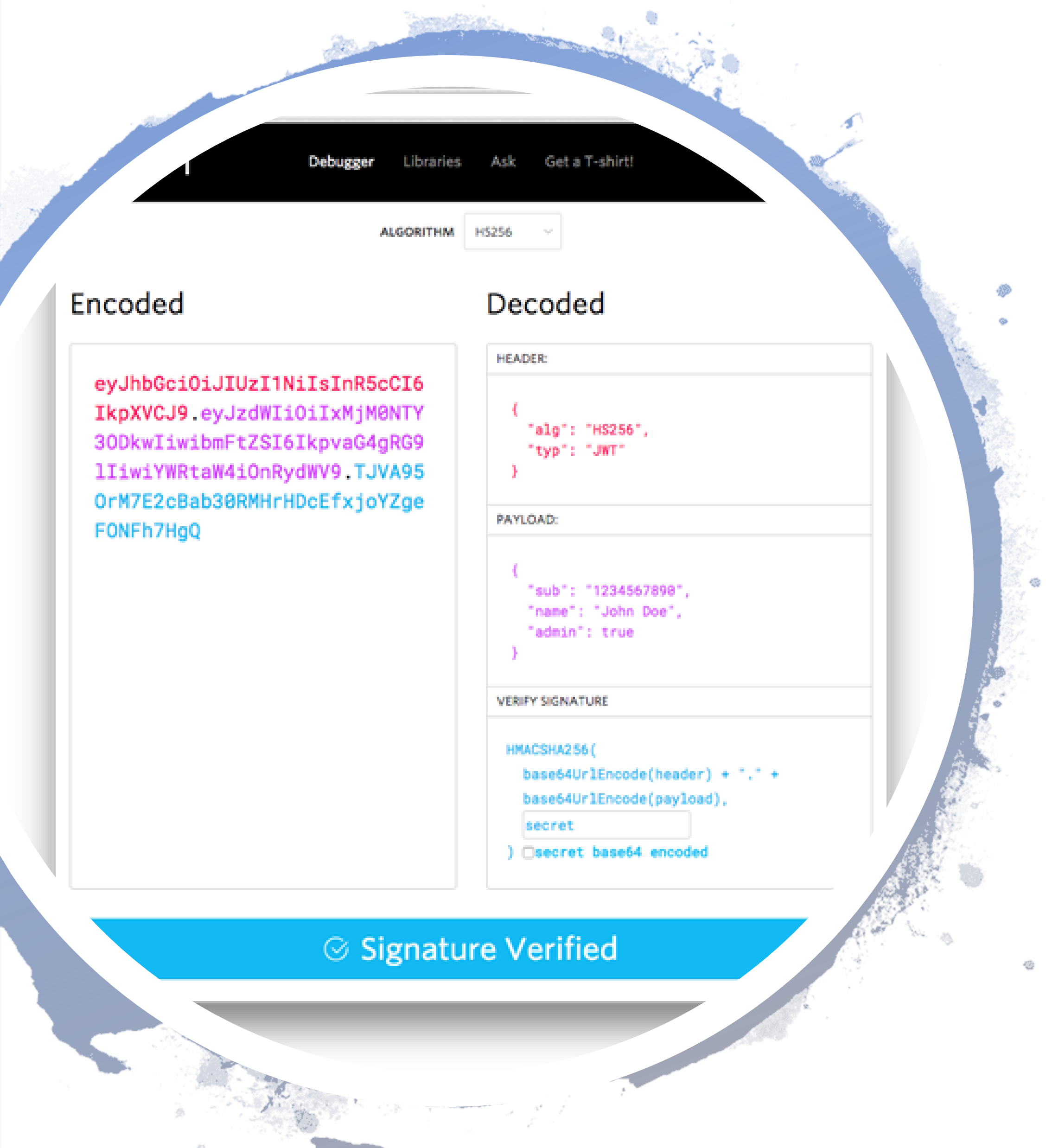
1	User **initiates** logout
2	Application **creates** logout request to Keycloak
2.1	Keycloak **returns** response logout to the application
3	Keycloak **creates** logout request to **another** application
3.1	Application **returns** response logout to Keycloak
4	Keycloak **terminates** session
5	Applications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



```
eyJhbGciOiJIUzI1NiIsInR5cCI6
IkpXVCJ9.eyJzdWIiOiIxMjM0NTY
3ODkwIiwibmFtZSI6IkpvaG4gRG9
lIiwiYWRtaW4iOnRydWV9.TJVA95
OrM7E2cBab30RMHrHDcEfxjoYZge
FONFh7HgQ
```

HEADER:

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

PAYLOAD:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

VERIFY SIGNATURE

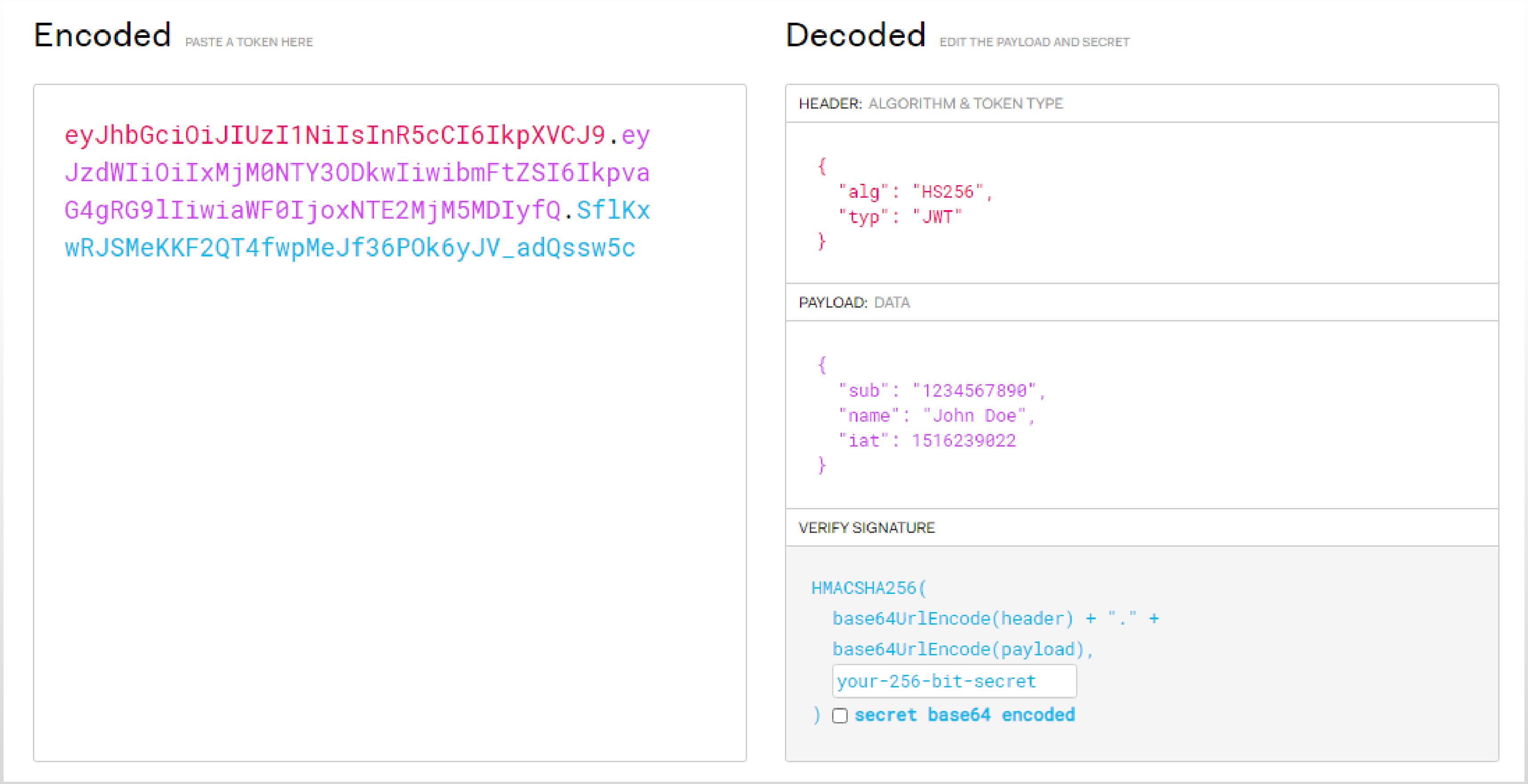
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) ☐secret base64 encoded
```

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.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_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
```

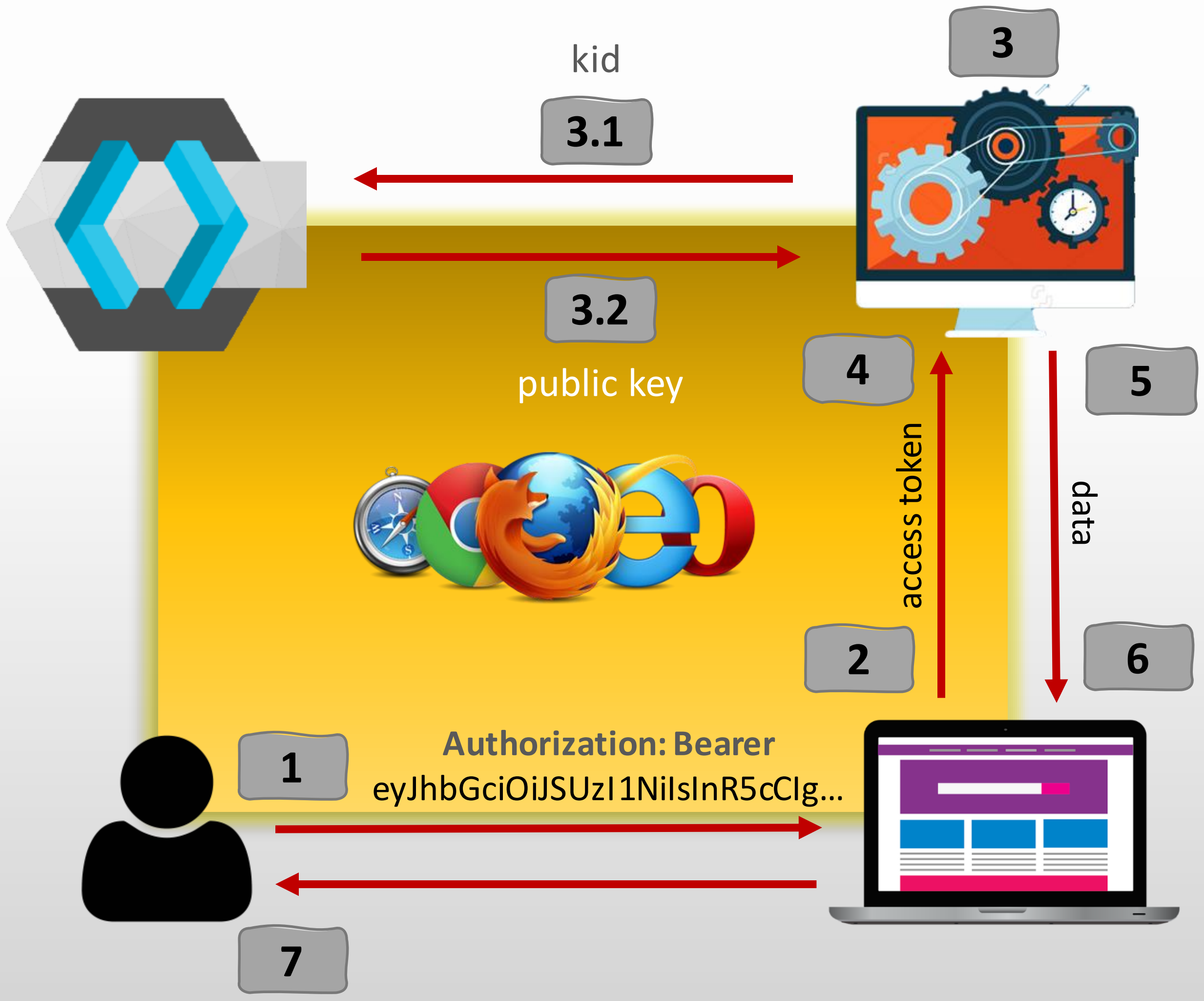
Note

Base64 means **Encoding**

Encoding != **Encryption**

Calling Backend Services

Calling Backend Services



1	Authenticated use **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

JBoss®
by Red Hat

SAML

Java
Servlets

WildFly

node
JS®

jetty://

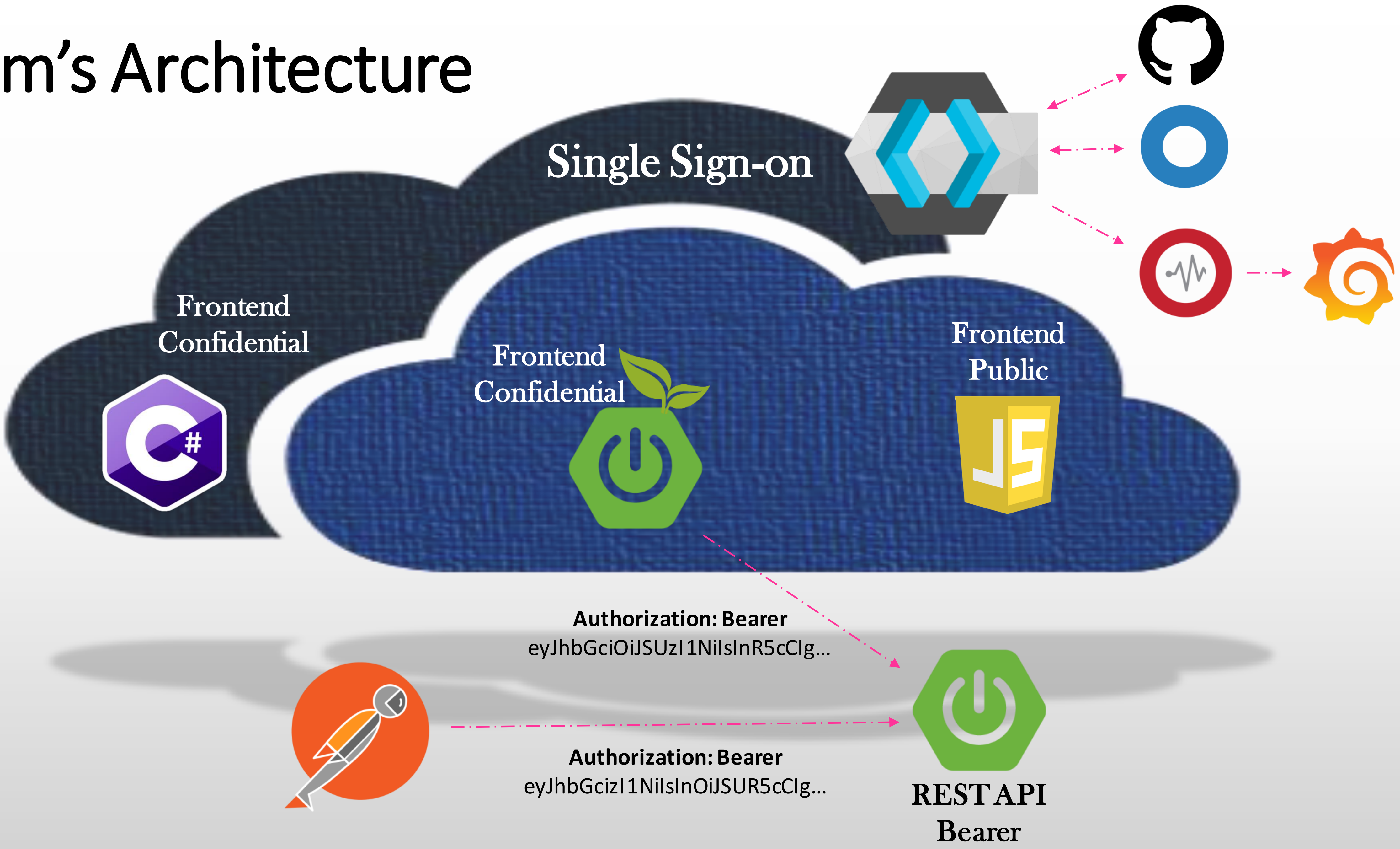
APACHE
HTTP SERVER

iOS

I DON'T KNOW RICK
LOOKS FAKE TO ME

LIVE DEMO

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

Twitter: @edlask8