Single Sign-on with KEYCLOAK

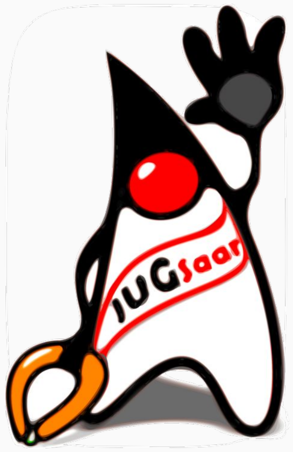
Thomas Darimont
eurodata AG

24.10.2017

Thomas Darimont

@thomasdarimont
@jugsaar

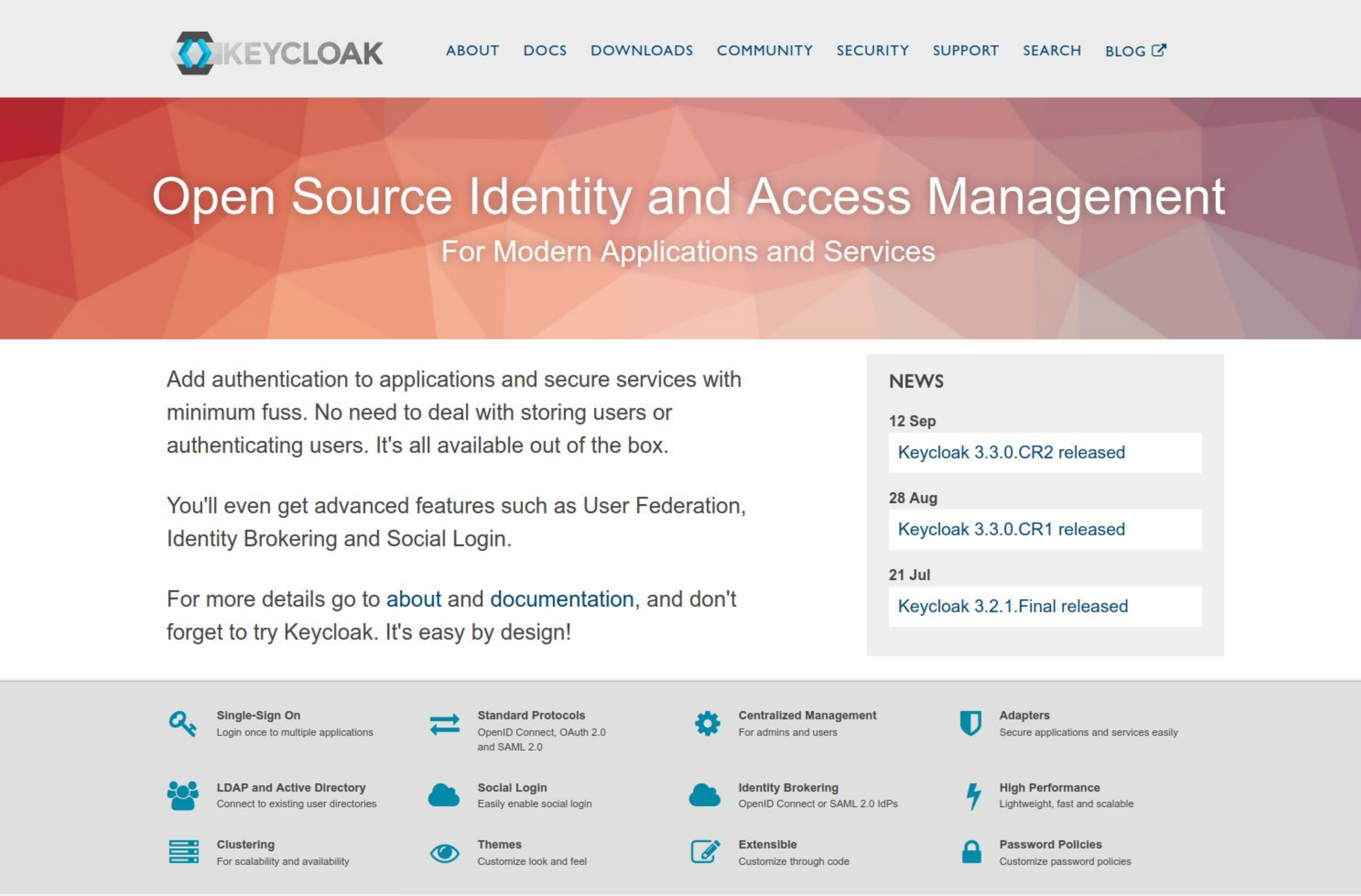
- Software Architect @ >eurodata
- Spring Team Alumni
- Open Source Enthusiast
- Java User Group Saarland Organizer
- Keycloak Contributor for over 2 years



The Journey

- Keycloak
- Single Sign-on
- Securing Applications
- Keycloak Extensions

Keycloak



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

12 Sep
Keycloak 3.3.0.CR2 released

28 Aug
Keycloak 3.3.0.CR1 released

21 Jul
Keycloak 3.2.1.Final released

Single-Sign On Login once to multiple applications	**Standard Protocols** OpenID Connect, OAuth 2.0 and SAML 2.0	**Centralized Management** For admins and users	**Adapters** Secure applications and services easily
LDAP and Active Directory Connect to existing user directories	**Social Login** Easily enable social login	**Identity Brokering** OpenID Connect or SAML 2.0 IdPs	**High Performance** Lightweight, fast and scalable
Clustering For scalability and availability	**Themes** Customize look and feel	**Extensible** Customize through code	**Password Policies** Customize password policies

keycloak.org

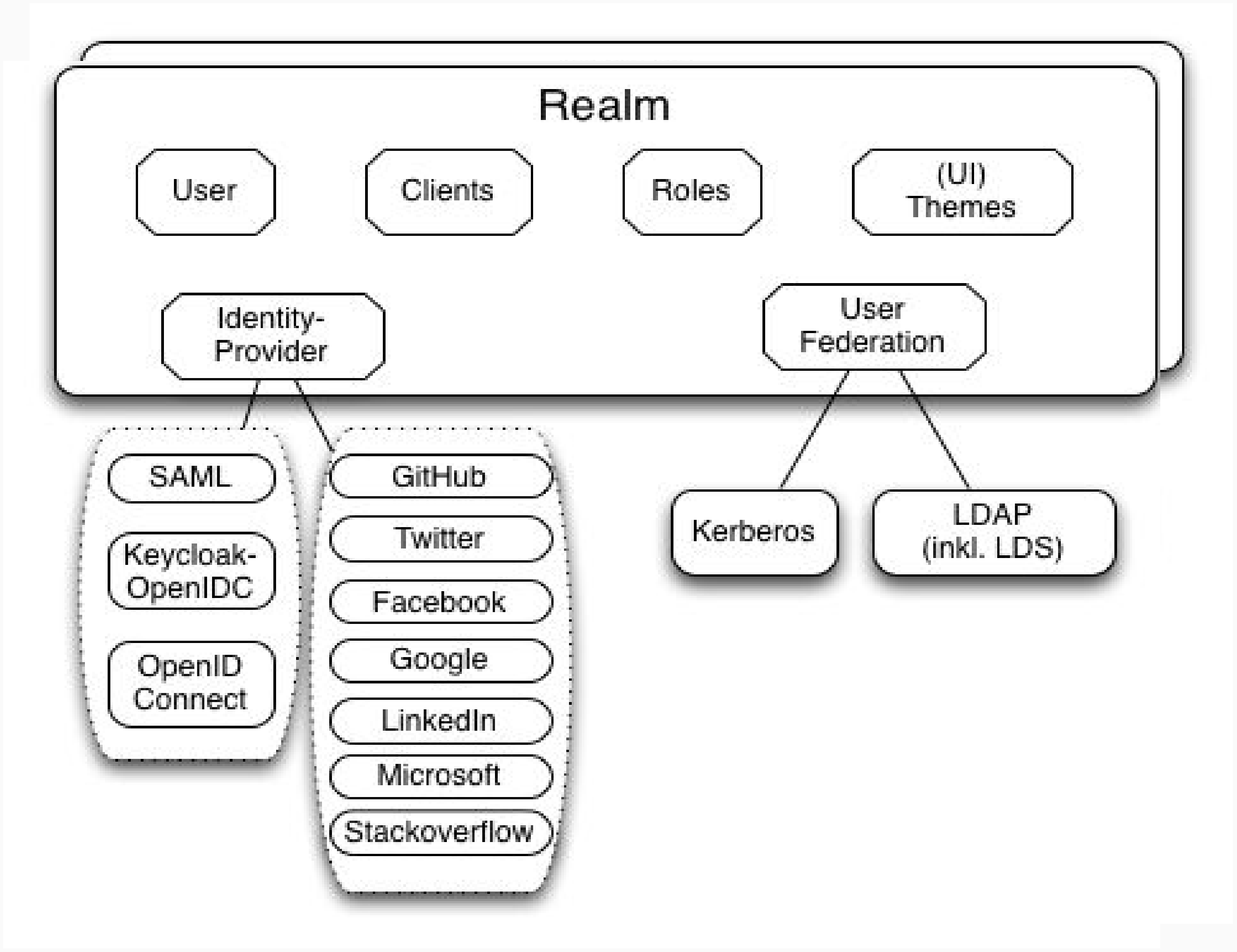
Keycloak Project

- **Open Source** Identity and Access Management Solution
- JBoss Developers (Red Hat)
- Since 2013, Release ~ every 6 Weeks
- Current Version 3.3.0.CR2
- Good documentation, many Examples
- Hosted on Github 208+ Contributors, 965+ Forks
- **Vital Community**
- **Commercial Offering** available

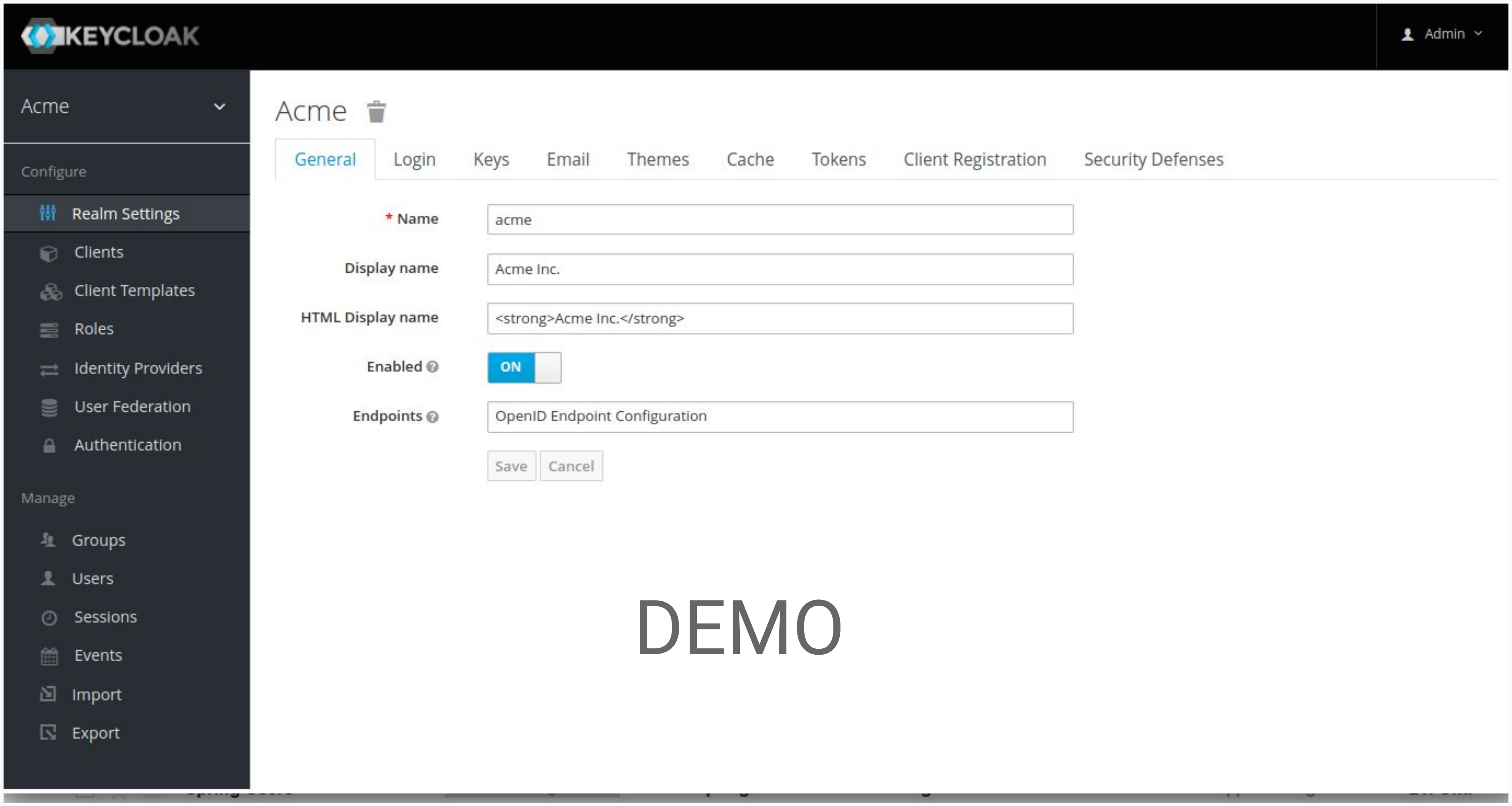
Keycloak Features

- **Single Sign-on** and Single Sign-out
- Flexible **Authentication** and **Authorization**
- **Multi-Factor Authentication** One-time Password
- **Standard Protocols** OAuth 2.0, OIDC 1.0, SAML 2.0, Docker Auth
- **Social Login** Google, Facebook, Twitter,...
- Provides centralized **User Management**
- Supports **Directory Services**
- **Customizable** and **Extensible**
- **Easy** Setup and Integration

Main Concepts



Admin Console



* Name	acme
Display name	Acme Inc.
HTML Display name	<strong>Acme Inc.</strong>
Enabled	ON
Endpoints	OpenID Endpoint Configuration

DEMO

Admin Console Login

Technology Stack - Keycloak 3.3.x

Admin Console

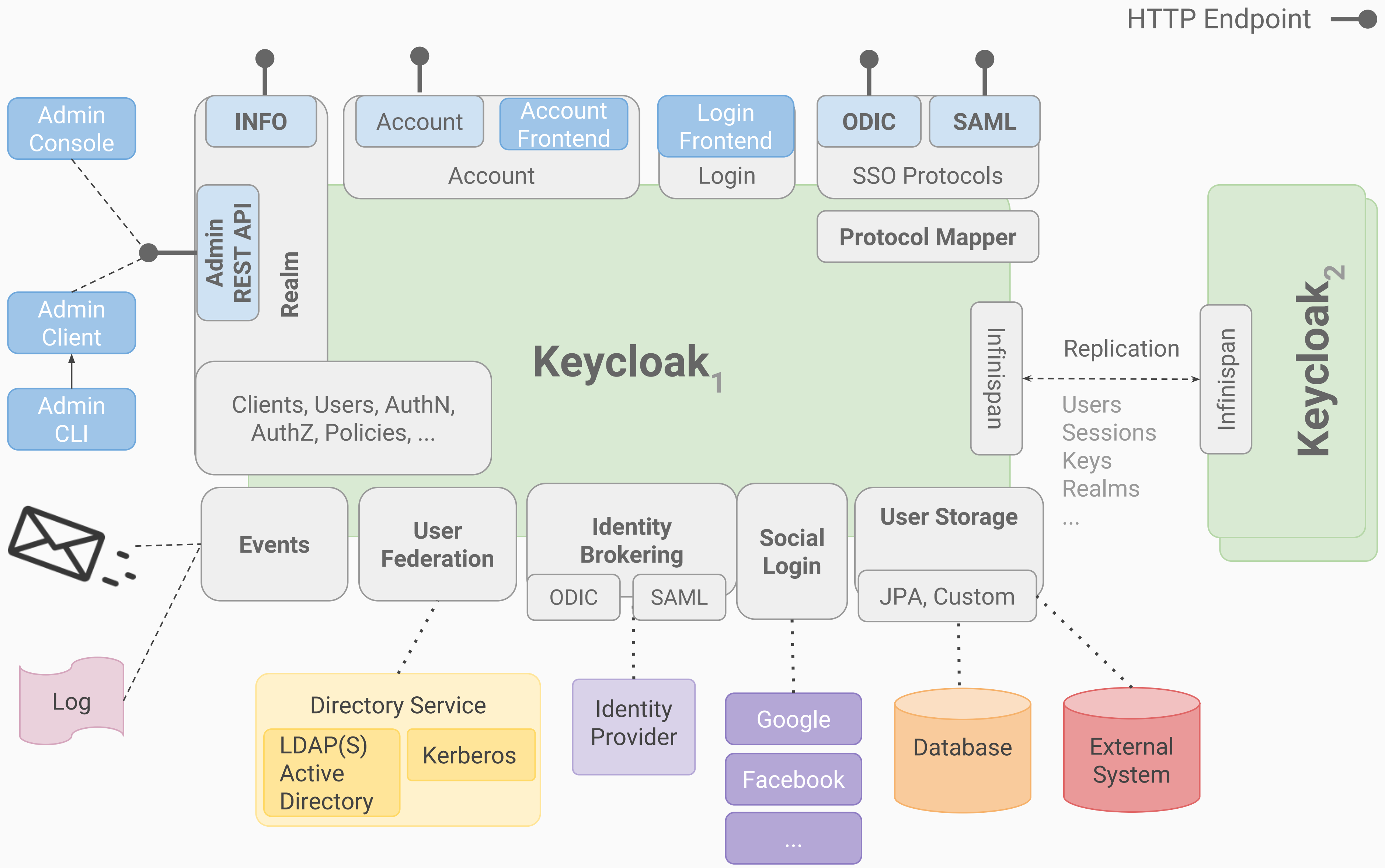
- Angular JS (1.6.4)
- PatternFly
- Bootstrap

↕

Server

- Wildfly 11.0.0.x
- JAX-RS (Resteasy)
- JPA (Hibernate)
- Infinispan (JGroups)
- Freemarker
- Jackson 2.0
- JBoss Logging
- Apache Directory API
- Commons HTTP Client

Server Architecture



Authentication & Authorization in Keycloak

- **Authentication (AuthN)**
  - Determines ***who** the user **is***
  - via *OIDC, SAML, Docker Auth, Kerberos*
  - Internal & Federated User Storage (Kerberos, LDAP, Custom)
- **Authorization (AuthZ)**
  - Determines ***what** the user **is allowed** to do*
  - Role based Access Control (RBAC)
  - *Authorization Services*
    - Flexible Access Control Management
    - More Variants like ABAC, UBAC, CBAC supported

Single Sign-on

in Keycloak

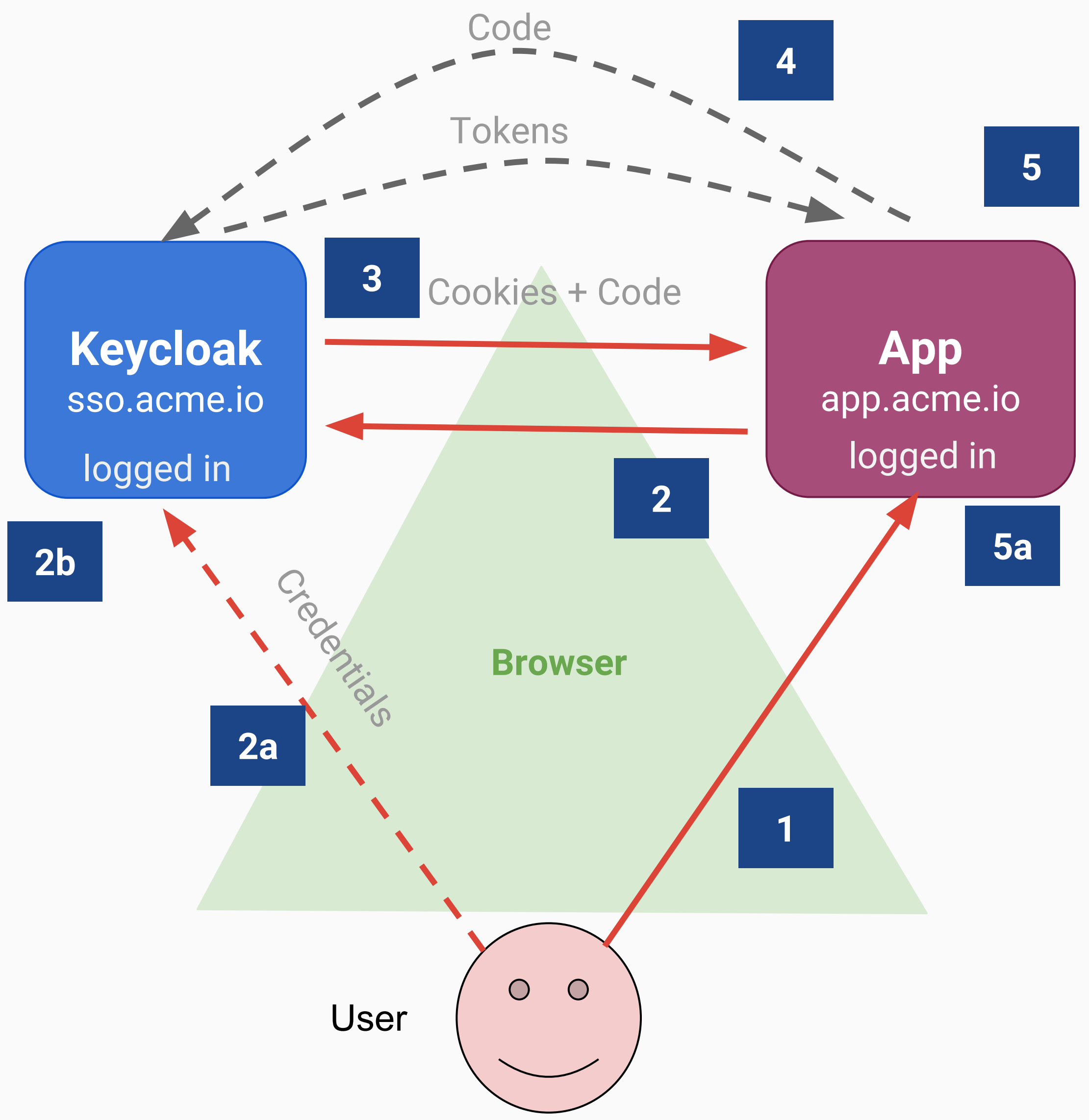
Single Sign-on & Single Sign-out

- **SSO** ⇒ Login **once** to access all applications
- **Standardized Protocols**
  - Open ID Connect 1.0 (OIDC)
  - Security Assertion Markup Language 2.0 (SAML)
- **Browser based "Web SSO"**
- works for Web, Mobile and Desktop Apps
- Support for **Single Sign-out**
  - Logouts can be propagated to clients
  - Clients can opt-in

Supported Single Sign-on Protocols

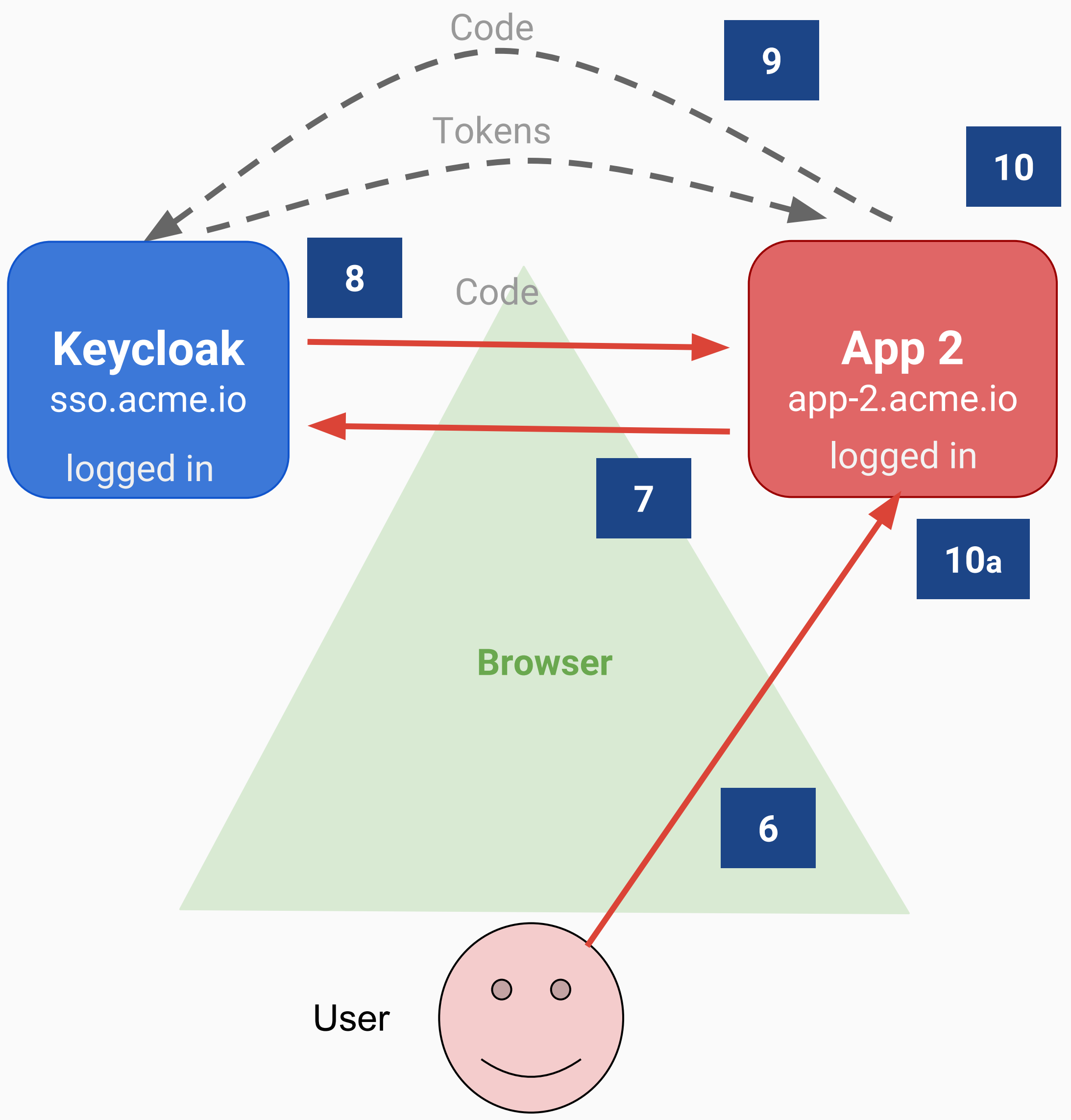
- **OpenID Connect 1.0**
  - Authentication protocol based on OAuth 2.0
  - Provides OAuth 2.0 tokens + IDToken to encode Identity
  - Tokens are encoded as JSON Web Tokens (JWT)
  - Requires communication over secure channel (HTTPS/TLS)
  - Recommended for Mobile- and Web-Applications
- **SAML 2.0** (Security Assertion Markup Language)
  - XML based authentication protocol
  - Uses XML signature and encryption → no secure channel required
  - Very mature standard & common in enterprise environments
  - Verbose and known to be not mobile friendly
- **Docker Registry v2 Authentication (new)**

Keycloak Web SSO with OIDC - Unauthenticated User



1	Unauthenticated User accesses App
2	App redirects to Keycloak for Login
2a	User submits Credentials to Keycloak
2b	Keycloak validates User Credentials
3	Keycloak creates SSO Session + Cookies and redirects User to App
4	App exchanges Code to Tokens with Keycloak via separate Channel
5	App verifies received Tokens and associates it with a session
5a	User is now “logged-in” to App

Keycloak Web SSO with OIDC - Authenticated User



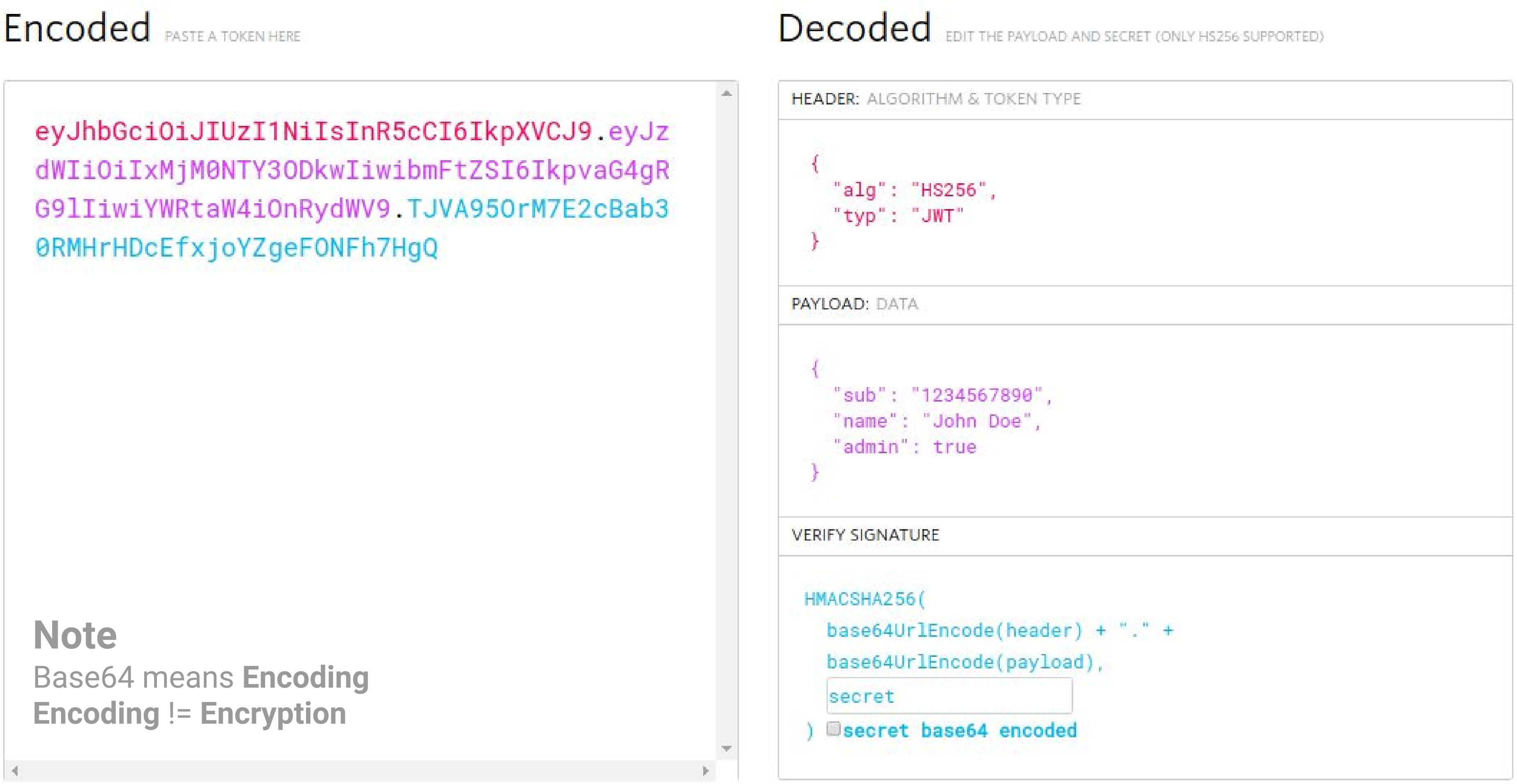
...	
6	Authenticated User accesses App 2
7	App 2 redirects User to Keycloak for Login
8	Keycloak detects SSO Session creates code & redirects to App 2
9	App 2 exchanges Code to Tokens with Keycloak via separate Channel
10	App 2 verifies received Tokens and associates it with a session
10a	User is now “logged-in” to App 2

Keycloak Tokens

- Token contains User information + Metadata
  - Signed self-contained **JSON Web Token** (JWT)
  - Issued by Keycloak, Signed with Realm Private Key
  - Limited lifespan, can be revoked
- Tokens can be verified by Clients
  - … by checking the Signature with Realm Public Key
  - … or via a HTTP POST to Keycloaks /token_introspection_endpoint
- Multiple Token Types
  - **`AccessToken`** short-lived (Minutes), used for accessing a Resource
  - **`IDToken`** contains information about User (OpenID Connect)
  - **`RefreshToken`** long-lived (Days), used for requesting new Tokens
  - **`OfflineToken`** special RefreshToken that “never” expires

JSON Web Token

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



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gR
G9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab3
0RMHrHDcEfxjoYZgeFONFh7HgQ
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

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

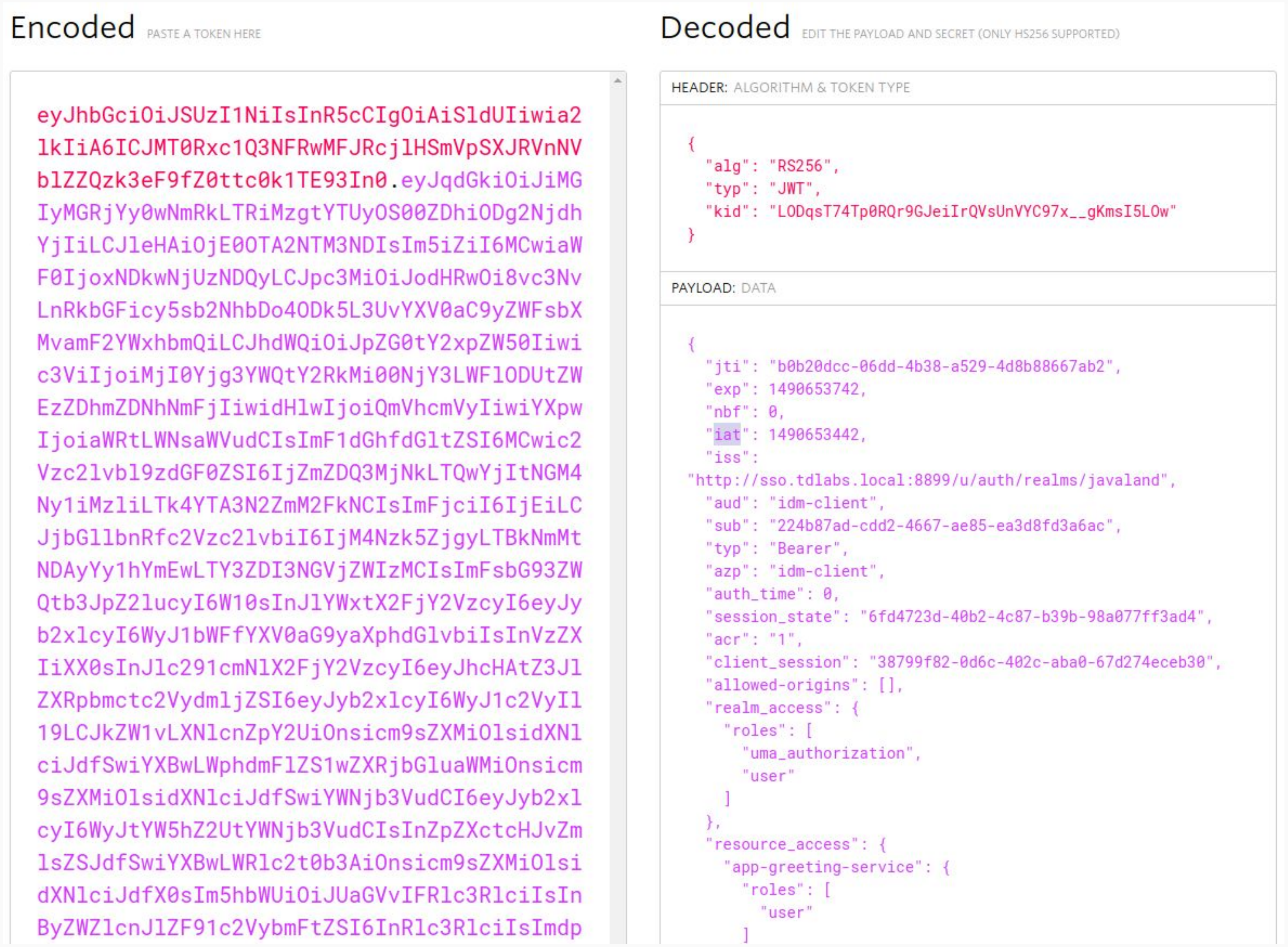
VERIFY SIGNATURE

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

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

https://jwt.io

Keycloak JWT Example



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2
lkIiA6ICJMT0Rxc1Q3NFRwMFJRcjlHSmVpSXJRVnNV
blZZQzk3eF9fZ0ttc0k1TE93In0.eyJqdGkiOiJiMG
IyMGRjYy0wNmRkLTRiMzgtYTUyOS00ZDhiODg2Njdh
YjIiLCJleHAiOjE0OTA2NTM3NDIsIm5iZiI6MCwiaW
F0IjoxNDkwNjUzNDQyLCJpc3MiOiJodHRwOi8vc3Nv
LnRkbGFicy5sb2NhbDo4ODk5L3UvYXV0aC9yZWFsbX
MvamF2YWxhbmQiLCJhdWQiOiJpZG0tY2xpZW50Iiwi
c3ViIjoiMjI0Yjg3YWQtY2RkMi00NjY3LWFlODUtZW
EzZDhmZDNhNmFjIiwidHlwIjoiQmVhcmVyIiwiYXpw
IjoiaWRtLWNsaWVudCIsImF1dGhfdGltZSI6MCwic2
Vzc2lvbl9zdGF0ZSI6IjZmZDQ3MjNkLTQwYjItNGM4
Ny1iMzliLTk4YTA3N2ZmM2FkNCIsImFjciI6IjEiLC
JjbGllbnRfc2Vzc2lvbiI6IjM4Nzk5ZjgyLTBkNmMt
NDAyYy1hYmEwLTY3ZDI3NGVjZWIzMCIsImFsbG93ZW
Qtb3JpZ2lucyI6W10sInJlYWxtX2FjY2VzcyI6eyJy
b2xlcyI6WyJ1bWFfYXV0aG9yaXphdGlvbiIsInVzZX
IiXX0sInJlc291cmNlX2FjY2VzcyI6eyJhcHAtZ3Jl
ZXRpbmctc2VydmljZSI6eyJyb2xlcyI6WyJ1c2VyIl
19LCJkZW1vLXNlcnZpY2UiOnsicm9sZXMiOlsidXNl
ciJdfSwiYXBwLWphdmFlZS1wZXRjbGluaWMiOnsicm
9sZXMiOlsidXNlciJdfSwiYWNjb3VudCI6eyJyb2xl
cyI6WyJtYW5hZ2UtYWNjb3VudCIsInZpZXctcHJvZm
lsZSJdfSwiYXBwLWRlc2t0b3AiOnsicm9sZXMiOlsi
dXNlciJdfX0sIm5hbWUiOiJUaGVvIFRlc3RlciIsIn
ByZWZlcnJlZF91c2VybmFtZSI6InRlc3RlciIsImdp
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

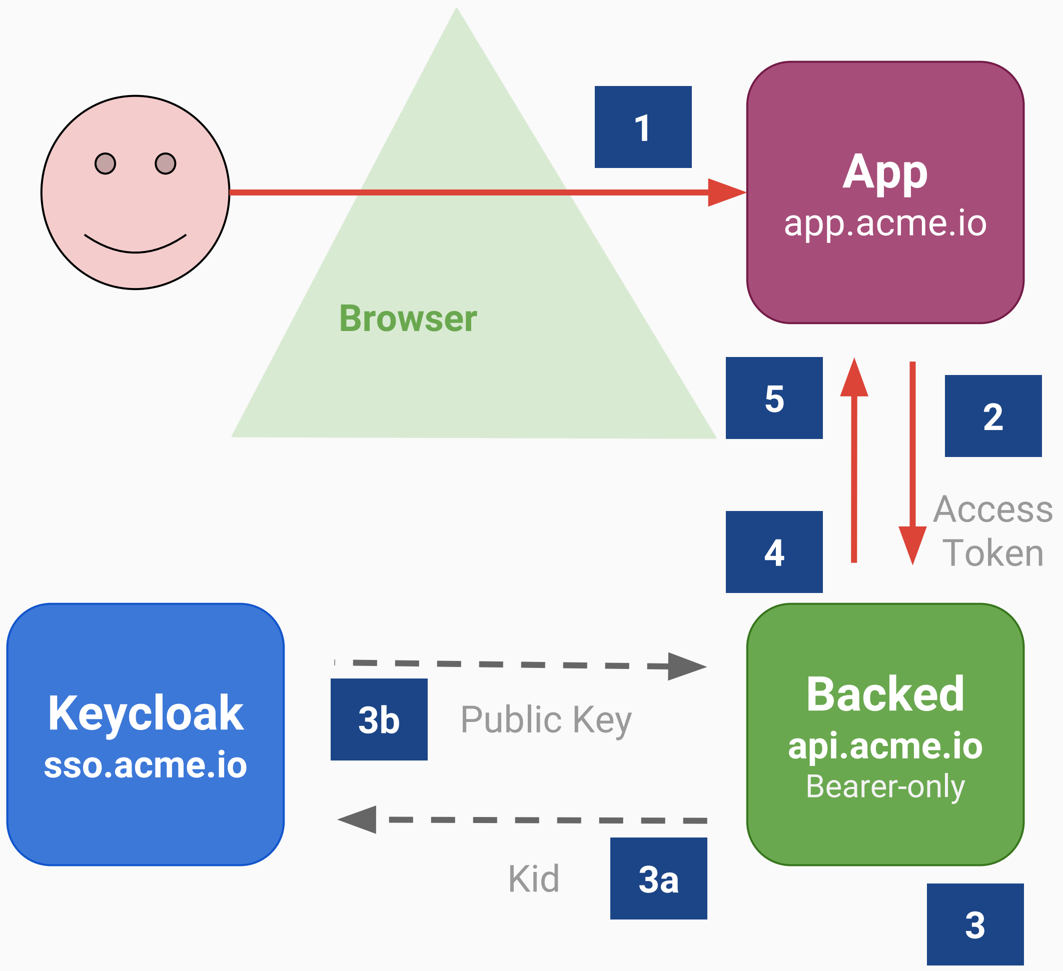
HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "LODqsT74Tp0RQr9GJeiIrQVsUnVYC97x__gKmsI5LOw"
}
```

PAYLOAD: DATA

```
{
  "jti": "b0b20dcc-06dd-4b38-a529-4d8b88667ab2",
  "exp": 1490653742,
  "nbf": 0,
  "iat": 1490653442,
  "iss":
"http://sso.tdlabs.local:8899/u/auth/realms/javaland",
  "aud": "idm-client",
  "sub": "224b87ad-cdd2-4667-ae85-ea3d8fd3a6ac",
  "typ": "Bearer",
  "azp": "idm-client",
  "auth_time": 0,
  "session_state": "6fd4723d-40b2-4c87-b39b-98a077ff3ad4",
  "acr": "1",
  "client_session": "38799f82-0d6c-402c-aba0-67d274eceb30",
  "allowed-origins": [],
  "realm_access": {
    "roles": [
      "uma_authorization",
      "user"
    ]
  },
  "resource_access": {
    "app-greeting-service": {
      "roles": [
        "user"
      ]
```

Calling Backend Services with AccessToken



1	Authenticated User accesses App
2	App uses AccessToken in HTTP Header to access backend
3	Backend looks-up Realm Public Key in cache with in Kid from JWT
3a	If not found, fetch Public Key with Kid from Keycloak
3b	Keycloak returns Realm Public Key
4	Backend verifies AccessToken Signature with Realm Public Key
5	Backend Service grants access and returns user data

Keycloak

Client Integration

Keycloak Integration Options

Keycloak Integrations

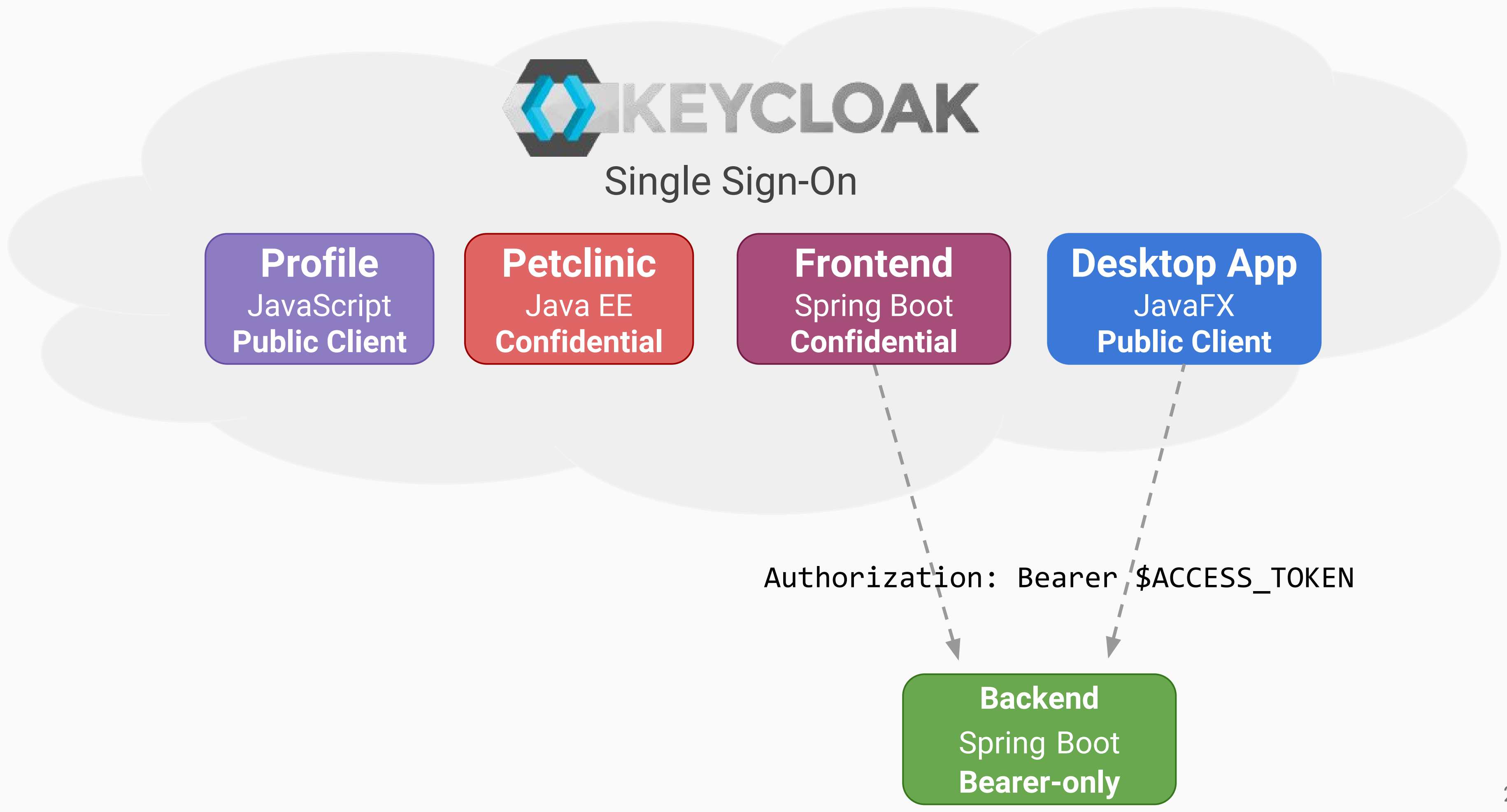
- OpenID Connect Adapters
  Spring Security, Spring Boot, ServletFilter, Tomcat, Jetty, Undertow, Wildfly, JBoss EAP, JAAS, …
  NodeJS, JavaScript, Angular, AngularJS, Aurelia, CLI & Desktop Apps...
- SAML Adapters
  ServletFilter, Tomcat, Jetty, Wildfly

Generic Integrations

- Apache Modules
  - `mod_auth_oidc` for OpenID Connect - maintained by Ping Identity
  - `mod_auth_mellon` for SAML - maintained by Red Hat
- Reverse Proxies
  - Official Keycloak Proxy injects auth info into HTTP headers
  - keycloak-proxy on github… same written in Go
- other Languages and Frameworks
  - Certified OpenID Connect Implementations
  - SAML Interoperable Implementations, Tools, Libraries and Services

Keycloak Demo
Securing Apps

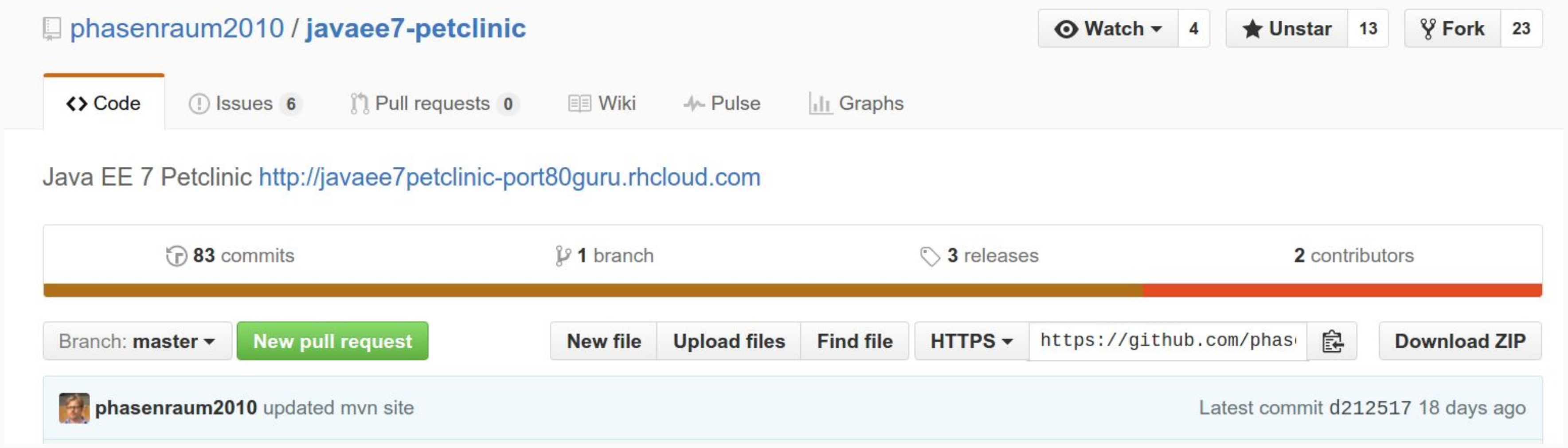
Demo Environment



Demo *Securing Apps*

Java EE 7 Petclinic

- Java EE 7 Web Application based on JSF, JAX-RS, JPA, Wildfly 10.0
- Integration via `keycloak-servlet-filter-adapter`
- https://github.com/jugsaar/javaee7-petclinic

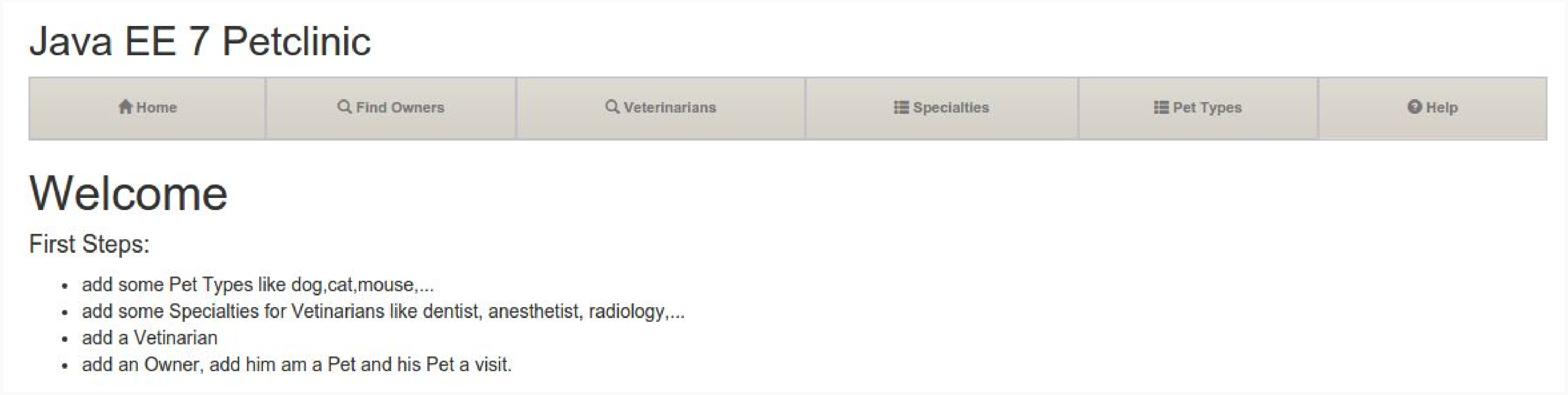

phasenraum2010 / javaee7-petclinic

Java EE 7 Petclinic http://javaee7petclinic-port80guru.rhcloud.com

83 commits · 1 branch · 3 releases · 2 contributors

phasenraum2010 updated mvn site — Latest commit d212517 18 days ago

Demo *Securing Apps* Before



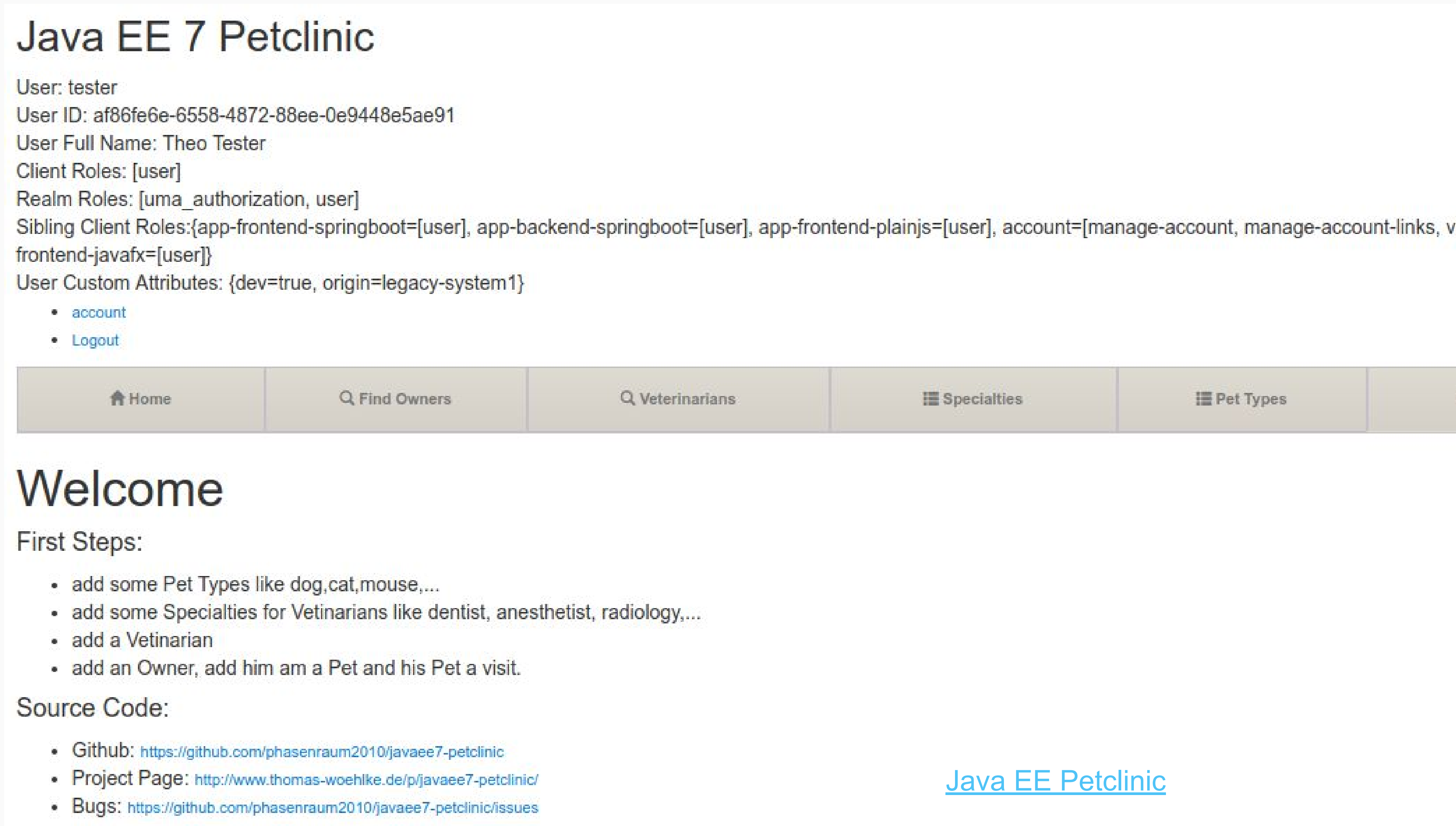
Java EE 7 Petclinic

Welcome

First Steps:

- add some Pet Types like dog,cat,mouse,...
- add some Specialties for Vetinarians like dentist, anesthetist, radiology,...
- add a Vetinarian
- add an Owner, add him am a Pet and his Pet a visit.

Demo *Securing Apps* After



Java EE 7 Petclinic

User: tester
User ID: af86fe6e-6558-4872-88ee-0e9448e5ae91
User Full Name: Theo Tester
Client Roles: [user]
Realm Roles: [uma_authorization, user]
Sibling Client Roles:{app-frontend-springboot=[user], app-backend-springboot=[user], app-frontend-plainjs=[user], account=[manage-account, manage-account-links, v
frontend-javafx=[user]}
User Custom Attributes: {dev=true, origin=legacy-system1}

- account
- Logout

Welcome

First Steps:

- add some Pet Types like dog,cat,mouse,...
- add some Specialties for Vetinarians like dentist, anesthetist, radiology,...
- add a Vetinarian
- add an Owner, add him am a Pet and his Pet a visit.

Source Code:

- Github: https://github.com/phasenraum2010/javaee7-petclinic
- Project Page: http://www.thomas-woehlke.de/p/javaee7-petclinic/
- Bugs: https://github.com/phasenraum2010/javaee7-petclinic/issues

[Java EE Petclinic]

Desktop Applications?

- **Two ways to integrate Desktop Applications**
  - Direct Access Grants - ***no*** SSO
  - KeycloakInstalled Adapter - SSO
- **Direct Access Grants**
  - Client sends HTTP POST request to Keycloaks /token Endpoint
    `client_id, username, password, grant_type=password`
  - Keycloak returns Tokens (Access, ID, Refresh)
  - Client needs to parse & validate tokens
  - Client sees password → Password Anti-Pattern
- **KeycloakInstalled Adapter**
  - Enables authorization code flow for Desktop / CLI apps
  - Code to token exchange via short lived `ServerSocket@localhost`
  - Uses Keycloak Login via Browser
  - Can reuse existing SSO session

Using the KeycloakInstalled Adapter

1 Add Maven Dependency

```xml
<dependency>
   <groupId>org.keycloak</groupId>
   <artifactId>keycloak-installed-adapter</artifactId>
   <version>${keycloak.version}</version>
</dependency>
```

2 Export keycloak.json for Client

```json
{ "realm": "acme",
 "auth-server-url": "http://sso.tdlabs.local:8899/u/auth",
 "ssl-required": "external",
 "resource": "app-frontend-javafx",
 "public-client": true,
 "use-resource-role-mappings": true }
```

3 Create KeycloakInstalled

```java
KeycloakInstalled keycloak = new KeycloakInstalled();
```

4 Trigger Browser login

```java
keycloak.loginDesktop();
```

5 Read current username

```java
keycloak.getIdToken().getPreferredUsername()
```

6 Read & use AccessToken string

```java
String token = keycloak.getTokenString(10, TimeUnit.SECONDS);
httpClient.header("Authorization", "Bearer " + token);
```

7 Trigger Browser Logout

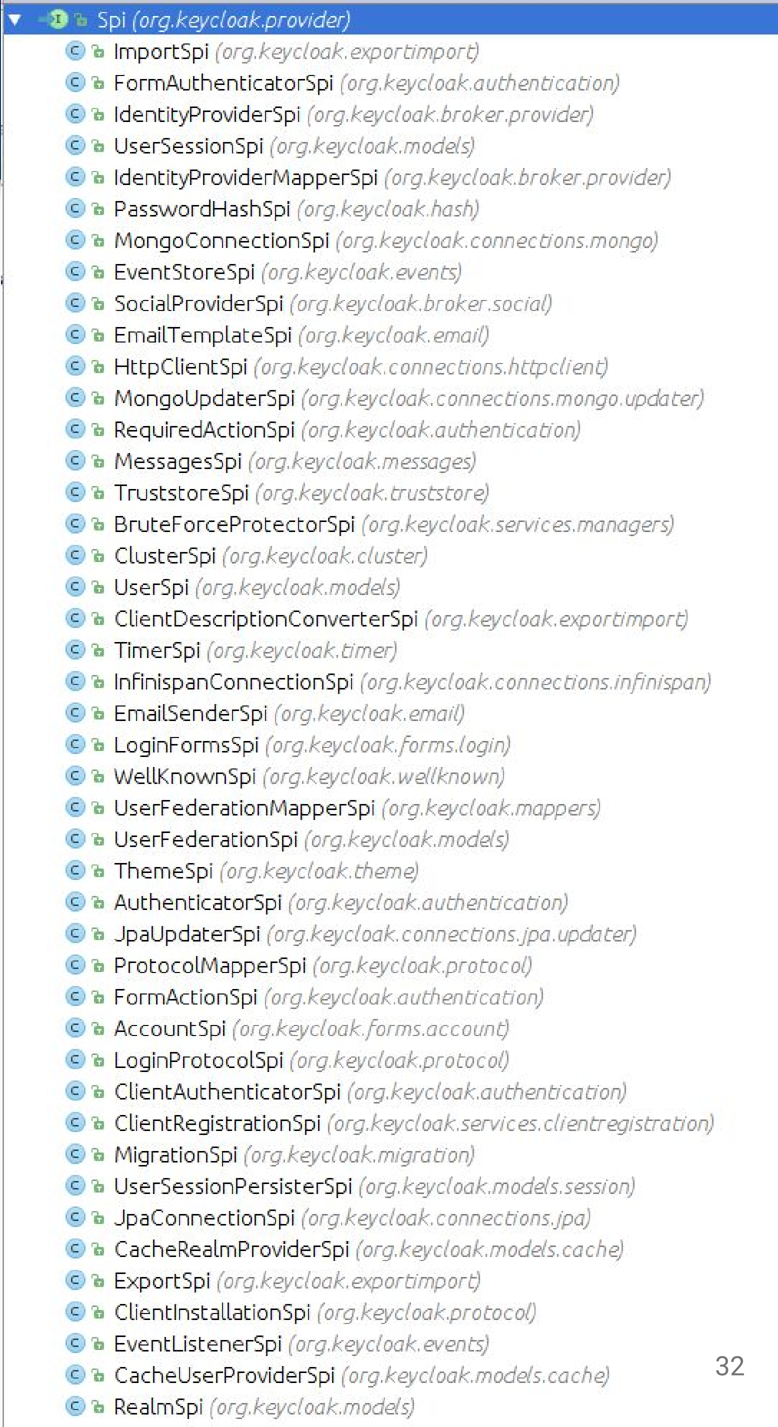
```java
keycloak.logout()
```

Keycloak

Extensions

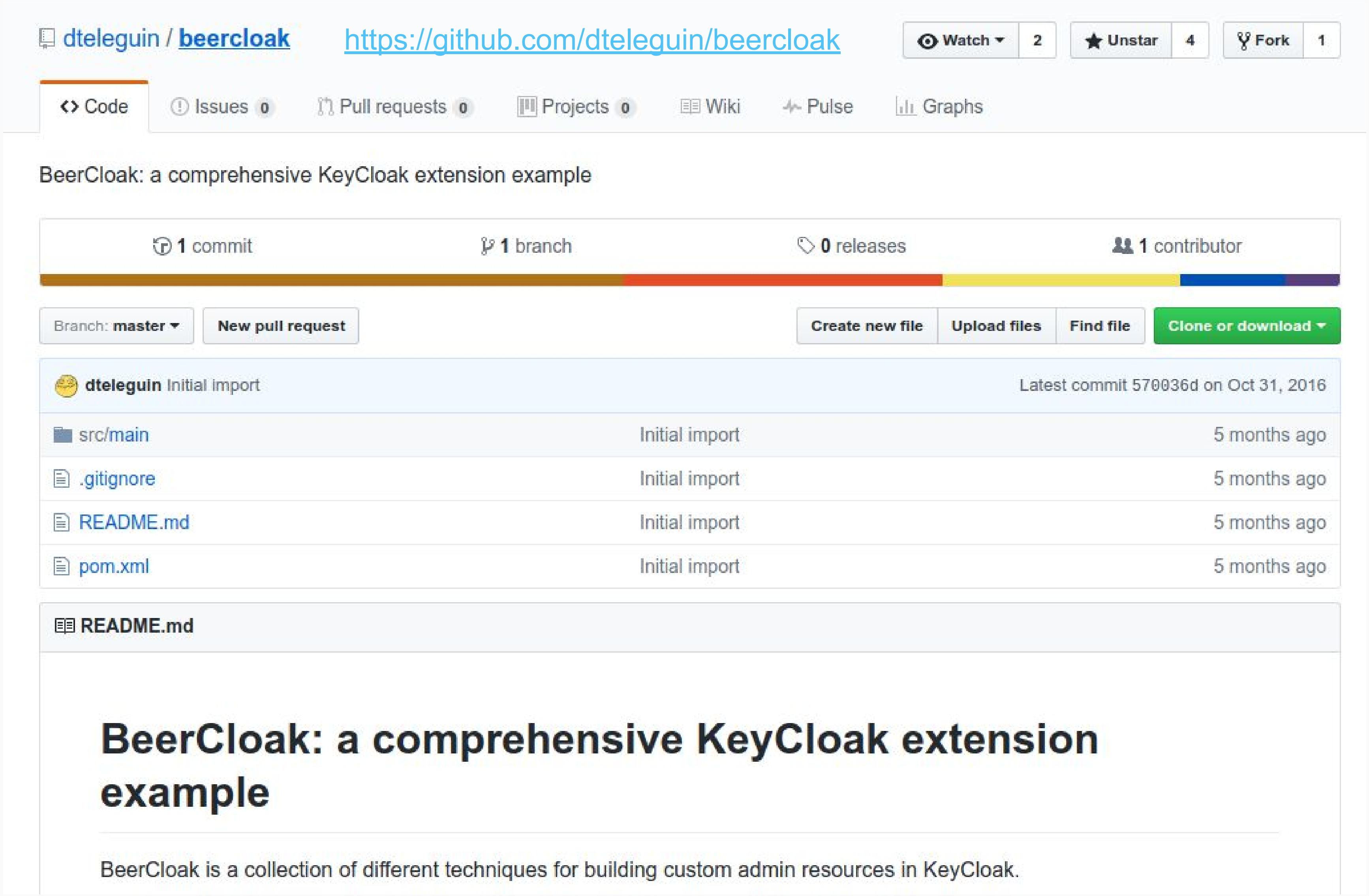
Keycloak Extension Points

- Extensions via Service Provider Interfaces
- Custom Authentication Mechanisms
- Custom “Required Actions”
- Custom User Storage (JDBC, REST, etc.)
- Event Listener (Provisioning, JMS)
- Credentials Hashing Mechanisms
- Custom REST Endpoints
- Custom Persistent Entities
- Custom Themes
- … many more



```
Spi (org.keycloak.provider)
  ImportSpi (org.keycloak.exportimport)
  FormAuthenticatorSpi (org.keycloak.authentication)
  IdentityProviderSpi (org.keycloak.broker.provider)
  UserSessionSpi (org.keycloak.models)
  IdentityProviderMapperSpi (org.keycloak.broker.provider)
  PasswordHashSpi (org.keycloak.hash)
  MongoConnectionSpi (org.keycloak.connections.mongo)
  EventStoreSpi (org.keycloak.events)
  SocialProviderSpi (org.keycloak.broker.social)
  EmailTemplateSpi (org.keycloak.email)
  HttpClientSpi (org.keycloak.connections.httpclient)
  MongoUpdaterSpi (org.keycloak.connections.mongo.updater)
  RequiredActionSpi (org.keycloak.authentication)
  MessagesSpi (org.keycloak.messages)
  TruststoreSpi (org.keycloak.truststore)
  BruteForceProtectorSpi (org.keycloak.services.managers)
  ClusterSpi (org.keycloak.cluster)
  UserSpi (org.keycloak.models)
  ClientDescriptionConverterSpi (org.keycloak.exportimport)
  TimerSpi (org.keycloak.timer)
  InfinispanConnectionSpi (org.keycloak.connections.infinispan)
  EmailSenderSpi (org.keycloak.email)
  LoginFormsSpi (org.keycloak.forms.login)
  WellKnownSpi (org.keycloak.wellknown)
  UserFederationMapperSpi (org.keycloak.mappers)
  UserFederationSpi (org.keycloak.models)
  ThemeSpi (org.keycloak.theme)
  AuthenticatorSpi (org.keycloak.authentication)
  JpaUpdaterSpi (org.keycloak.connections.jpa.updater)
  ProtocolMapperSpi (org.keycloak.protocol)
  FormActionSpi (org.keycloak.authentication)
  AccountSpi (org.keycloak.forms.account)
  LoginProtocolSpi (org.keycloak.protocol)
  ClientAuthenticatorSpi (org.keycloak.authentication)
  ClientRegistrationSpi (org.keycloak.services.clientregistration)
  MigrationSpi (org.keycloak.migration)
  UserSessionPersisterSpi (org.keycloak.models.session)
  JpaConnectionSpi (org.keycloak.connections.jpa)
  CacheRealmProviderSpi (org.keycloak.models.cache)
  ExportSpi (org.keycloak.exportimport)
  ClientInstallationSpi (org.keycloak.protocol)
  EventListenerSpi (org.keycloak.events)
  CacheUserProviderSpi (org.keycloak.models.cache)
  RealmSpi (org.keycloak.models)
```

Keycloak Extensions: BeerCloak



dteleguin / **beercloak**

https://github.com/dteleguin/beercloak

BeerCloak: a comprehensive KeyCloak extension example

dteleguin Initial import — Latest commit 570036d on Oct 31, 2016

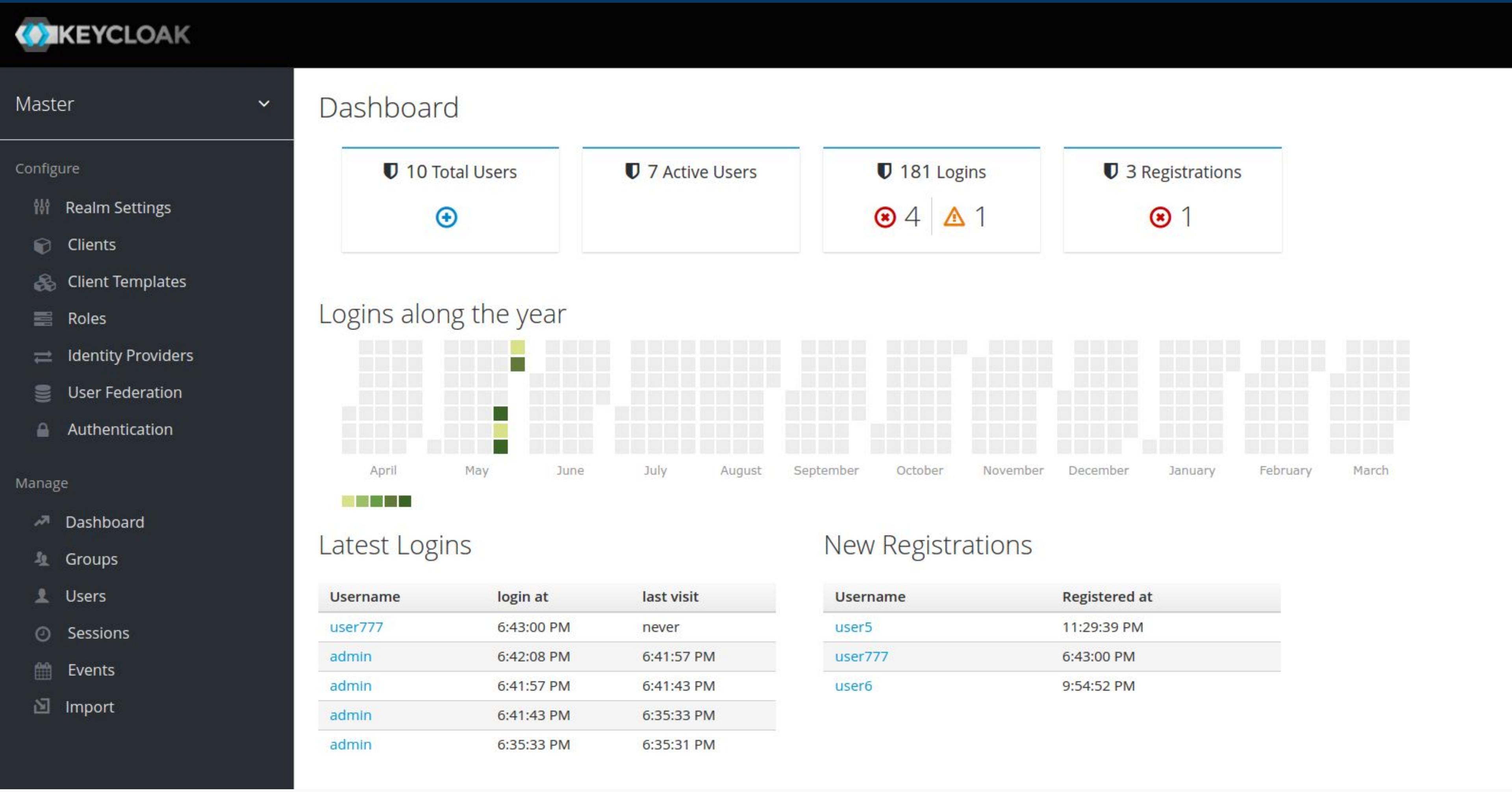
src/main	Initial import	5 months ago
.gitignore	Initial import	5 months ago
README.md	Initial import	5 months ago
pom.xml	Initial import	5 months ago

README.md

BeerCloak: a comprehensive KeyCloak extension example

BeerCloak is a collection of different techniques for building custom admin resources in KeyCloak.

Custom Dashboard Extension



Dashboard

10 Total Users

7 Active Users

181 Logins 4 1

3 Registrations 1

Logins along the year

Latest Logins

Username	login at	last visit
user777	6:43:00 PM	never
admin	6:42:08 PM	6:41:57 PM
admin	6:41:57 PM	6:41:43 PM
admin	6:41:43 PM	6:35:33 PM
admin	6:35:33 PM	6:35:31 PM

New Registrations

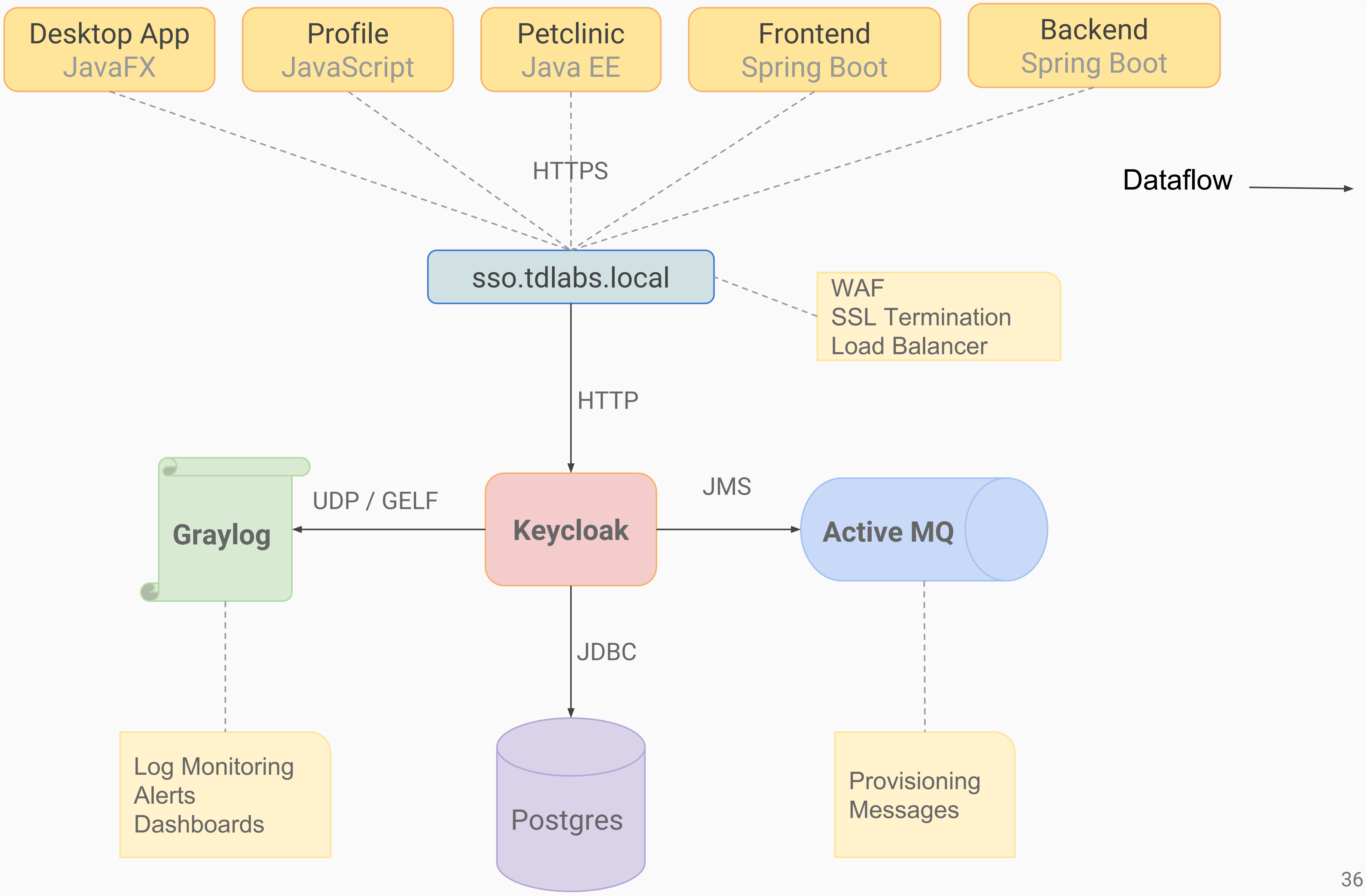
Username	Registered at
user5	11:29:39 PM
user777	6:43:00 PM
user6	9:54:52 PM

Please vote :) https://issues.jboss.org/browse/KEYCLOAK-1840

Keycloak

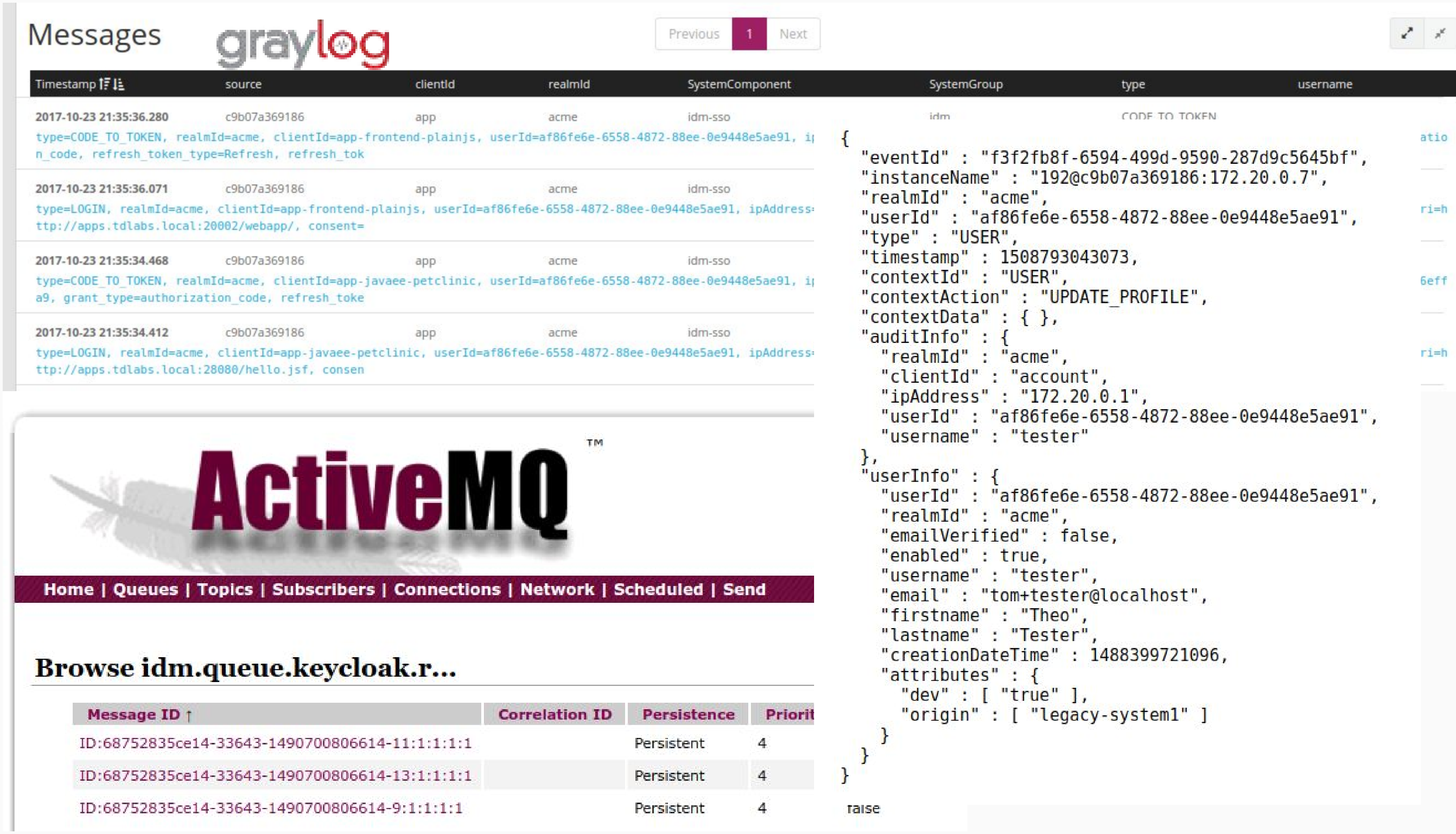
In the Field

Demo Environment (Docker)



https://github.com/thomasdarimont/jcon2017-keycloak

Example Docker Environment Demo


Messages
graylog
Previous
1
Next
ActiveMQ
TM
Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Timestamp	source	clientId	realmId	SystemComponent	SystemGroup	type	username
2017-10-23 21:35:36.280	c9b07a369186	app	acme	idm-sso	idm	CODE_TO_TOKEN	
2017-10-23 21:35:36.071	c9b07a369186	app	acme	idm-sso			
2017-10-23 21:35:34.468	c9b07a369186	app	acme	idm-sso			
2017-10-23 21:35:34.412	c9b07a369186	app	acme	idm-sso			

```
type=CODE_TO_TOKEN, realmId=acme, clientId=app-frontend-plainjs, userId=af86fe6e-6558-4872-88ee-0e9448e5ae91, i
n_code, refresh_token_type=Refresh, refresh_tok
type=LOGIN, realmId=acme, clientId=app-frontend-plainjs, userId=af86fe6e-6558-4872-88ee-0e9448e5ae91, ipAddress=
ttp://apps.tdlabs.local:20002/webapp/, consent=
type=CODE_TO_TOKEN, realmId=acme, clientId=app-javaee-petclinic, userId=af86fe6e-6558-4872-88ee-0e9448e5ae91, i
a9, grant_type=authorization_code, refresh_toke
type=LOGIN, realmId=acme, clientId=app-javaee-petclinic, userId=af86fe6e-6558-4872-88ee-0e9448e5ae91, ipAddress=
ttp://apps.tdlabs.local:28080/hello.jsf, consen
```

Browse idm.queue.keycloak.r...

Message ID ↑	Correlation ID	Persistence	Priorit
ID:68752835ce14-33643-1490700806614-11:1:1:1:1		Persistent	4
ID:68752835ce14-33643-1490700806614-13:1:1:1:1		Persistent	4
ID:68752835ce14-33643-1490700806614-9:1:1:1:1		Persistent	4

false

```
{
  "eventId" : "f3f2fb8f-6594-499d-9590-287d9c5645bf",
  "instanceName" : "192@c9b07a369186:172.20.0.7",
  "realmId" : "acme",
  "userId" : "af86fe6e-6558-4872-88ee-0e9448e5ae91",
  "type" : "USER",
  "timestamp" : 1508793043073,
  "contextId" : "USER",
  "contextAction" : "UPDATE_PROFILE",
  "contextData" : { },
  "auditInfo" : {
    "realmId" : "acme",
    "clientId" : "account",
    "ipAddress" : "172.20.0.1",
    "userId" : "af86fe6e-6558-4872-88ee-0e9448e5ae91",
    "username" : "tester"
  },
  "userInfo" : {
    "userId" : "af86fe6e-6558-4872-88ee-0e9448e5ae91",
    "realmId" : "acme",
    "emailVerified" : false,
    "enabled" : true,
    "username" : "tester",
    "email" : "tom+tester@localhost",
    "firstname" : "Theo",
    "lastname" : "Tester",
    "creationDateTime" : 1488399721096,
    "attributes" : {
      "dev" : [ "true" ],
      "origin" : [ "legacy-system1" ]
    }
  }
}
```

atio
ri=h
6eff
ri=h

Tips for working with Keycloak

- **Keep your Tokens small**
  - HTTP Header limits!
  - Only put in the token what you really need (*Full Scope Allowed = off*)
- **Keycloak provides a Realm-scoped Admin Console**
  - http://kc-host:8080/auth/admin/my-realm/console
  - Admin users needs permissions for `realm-management` in `my-realm`
- **Keycloak Admin CLI**
  - See **Blog Post**
  - `$KEYCLOAK_HOME/bin/kcadm.sh create users -r acme -s username=bubu`
- **Secure your Keycloak Installation!**
  - Keycloak exposes some undocumented Endpoints by default on server AND client!
  - Lock down `/admin`
  - Tip: Inspect other Keycloak instances to learn what to hide
    - Google Search for Keycloak Endpoints
    - Shodan search for Keycloak

Summary KEYCLOAK

- Easy to get started
  - unzip & run
- Provides many features out of the Box
  - SSO, Social Login, Federation, User Management,...
- Builds on proven and robust standards
  - OAuth 2.0, OpenID Connect 1.0, SAML 2.0
- Very extensible and easy to integrate
  - Many extension points & customization options
- a Pivotal part of an Identity Management infrastructure

Links

Keycloak Website

Keycloak Docs

Keycloak Blog

Keycloak User Mailing List

Keycloak Developer Mailing List

OpenID Connect

SAML

JSON Web Tokens

Awesome Keycloak

Keycloak Dockerized Examples

Keycloak Quickstarts Example Projects

Some Missing Features

Keycloak already provides a lot out of the box, but…

- Analyzing events in Admin Console is tedious and very limited
- Events don’t contain enough information
- No hooks to notify other applications about User Profile changes

Needed to extend Keycloak with...

- Custom `EventListener` that enriches and forwards events via JMS
- Integrated `GELF` Logging Appender to ship logs to Graylog Log Server
- See https://github.com/jugsaar/visit-yajug-20161023-keycloak

Accessing the API Backend with CURL

1 Request new Tokens via Password Credentials Grant (Direct Access Grants in Keycloak)

```
KC_RESPONSE=$(curl -X POST \
  http://sso.tdlabs.local:8899/u/auth/realms/acme/protocol/openid-connect/token \
  -d 'grant_type=password' \
  -d 'username=tester&password=test' \
  -d 'client_id=app-frontend-springboot' \
  -d 'client_secret=4822a740-20b9-4ff7-bbed-e664f4a70eb6' \
)
```

2 Extract AccessToken

```
KC_ACCESS_TOKEN=$(echo $KC_RESPONSE | jq -r .access_token)
# eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJGY3RMVHJqeWRxYkpISGZ0d29U ...
```

3 Use AccessToken in Authorization Header

```
curl \
  -H "Authorization: Bearer $KC_ACCESS_TOKEN" \
  http://apps.tdlabs.local:20000/todos/search/my-todos

Response
{
  "_embedded" : { "todos" : [ { "name" : "Buy milk",  "description" : "...",...
```