“Oh, we also need a Login Service”

The Keycloak project

AGENDA

What are we doing today?

Motivation
What, why, how?

OIDC
The Standard for web auth

Keycloak
The project

Using it
Features & Development

Running it
Ops

codecentric

EASY

SECURE

Thousand Passwords

"justanotherpassword12"

Design

"The value of VerifyPassword does not match pattern '.*[A-Z].*'"

Leaks

Managing Passwords is hard

No value added

"Lets do that later!"

How to improve?

- Use Standards
- Dont do it yourself

Authentication

Who are you?

Authorization

What are you allowed to do?

OpenID Connect

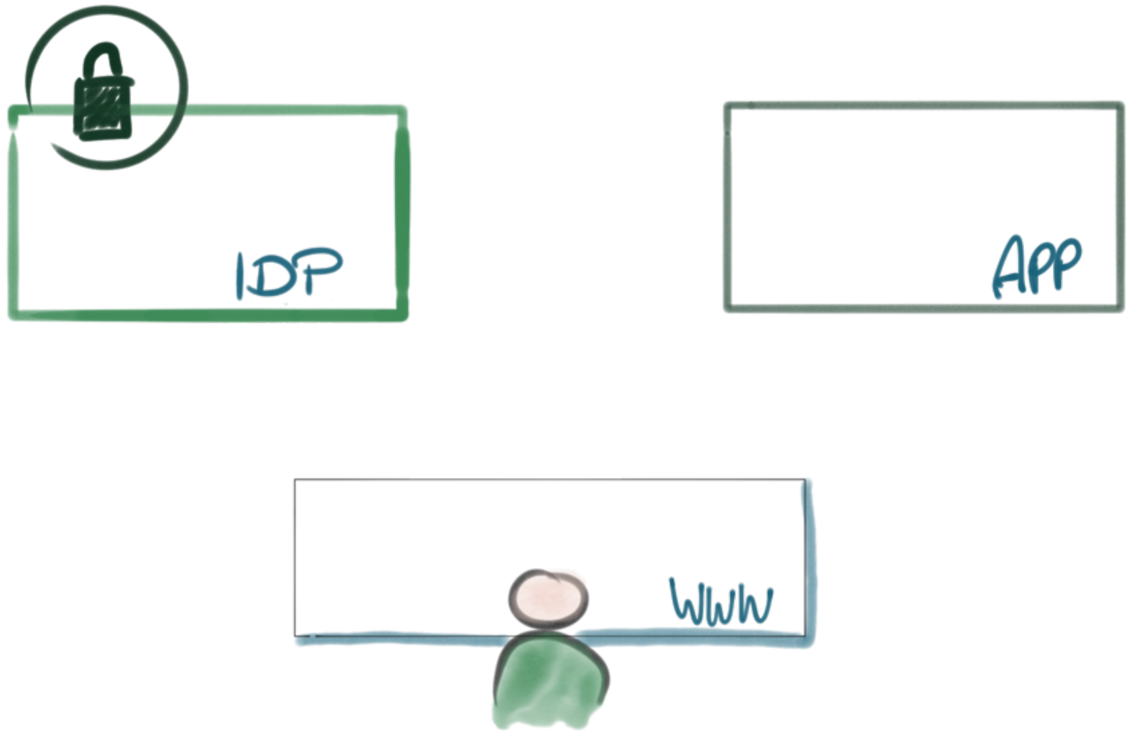
and OAuth2.0

How does these Standard Web SSO Protocols work?

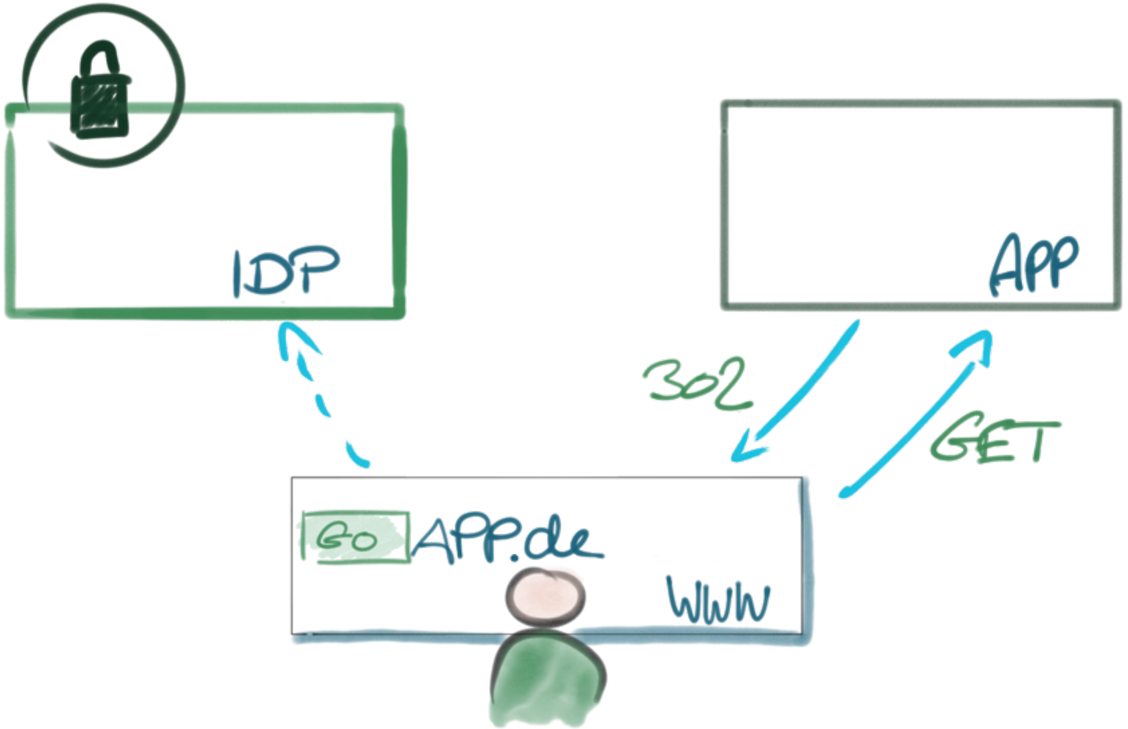
OpenID Connect

Authorization Code Flow Steps

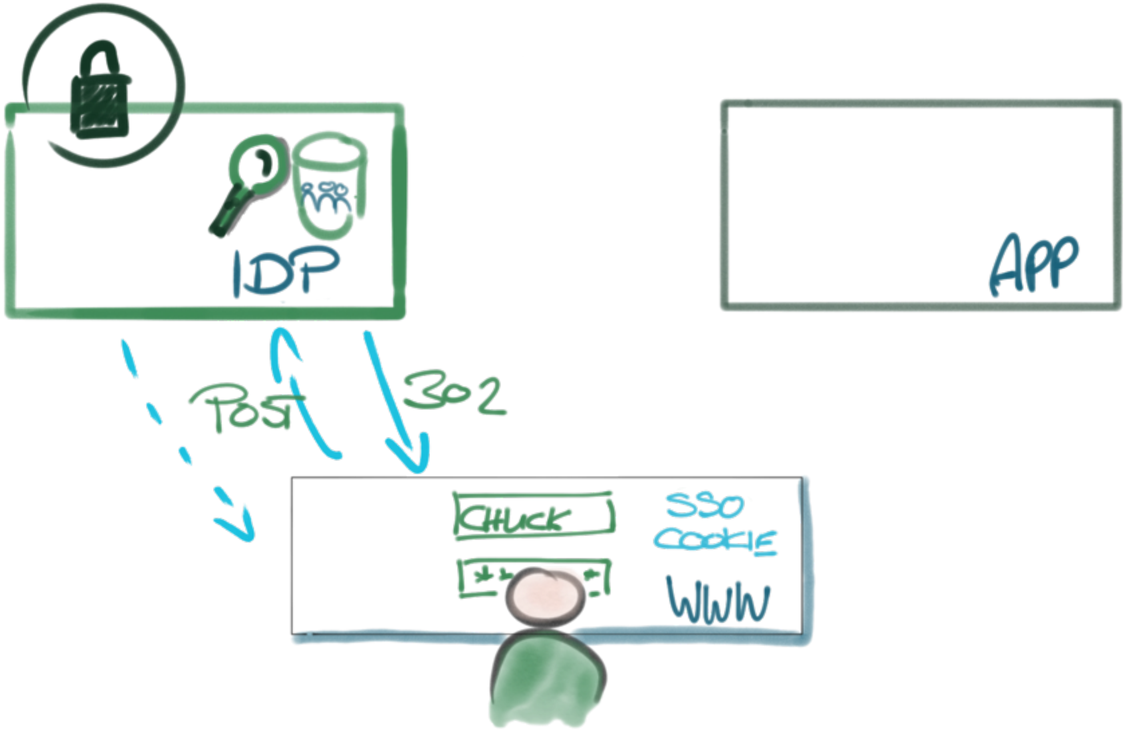
1. Client prepares an Authentication Request containing the desired request parameters.
2. Client sends the request to the Authorization Server.
3. Authorization Server Authenticates the End-User.
4. Authorization Server obtains End-User Consent/ Authorization.
5. Authorization Server sends the End-User back to the Client with an Authorization Code.
6. Client requests a response using the Authorization Code at the Token Endpoint.
7. Client receives a response that contains an ID Token and Access Token in the response body.
8. Client validates the ID token and retrieves the End-User's Subject Identifier.


IDP
APP
WWW

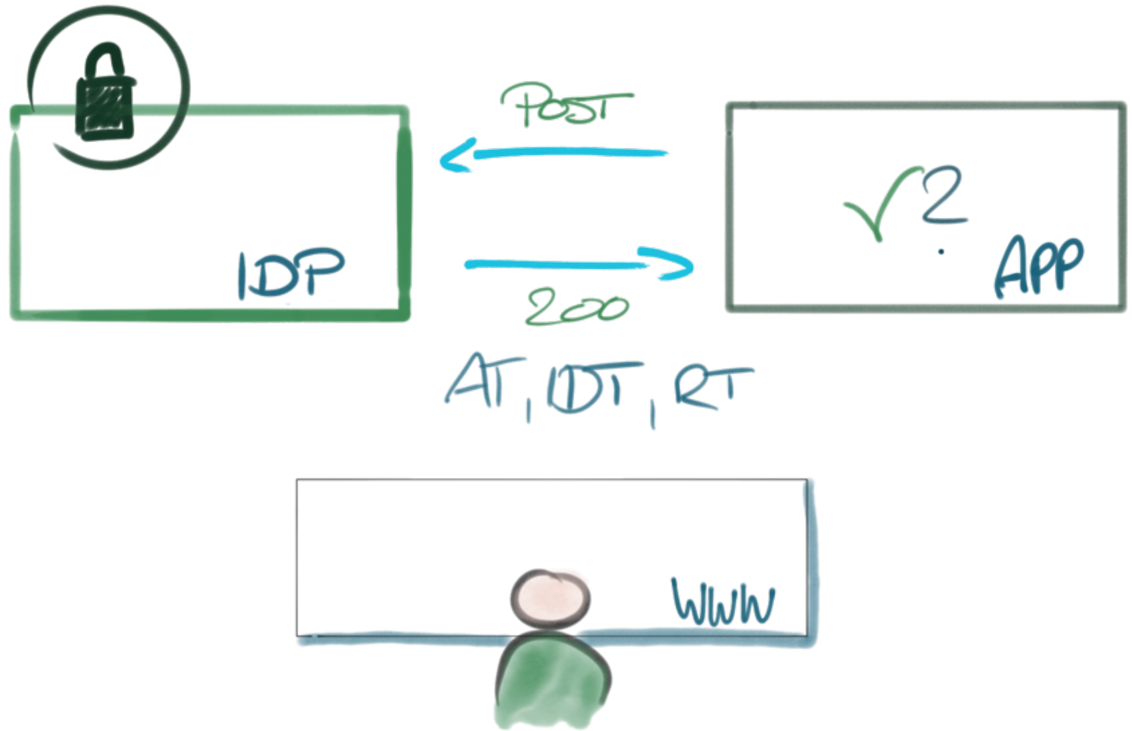
```
HTTP/1.1 302 Found
Location: https://idp.com/authorize?
  response_type=code
  &scope=openid%20profile%20email
  &client_id=s6BhdRkqt3
  &state=af0ifjsldkj
  &redirect_uri=https%3A%2F%2Fapp.deg%2Fcb
```



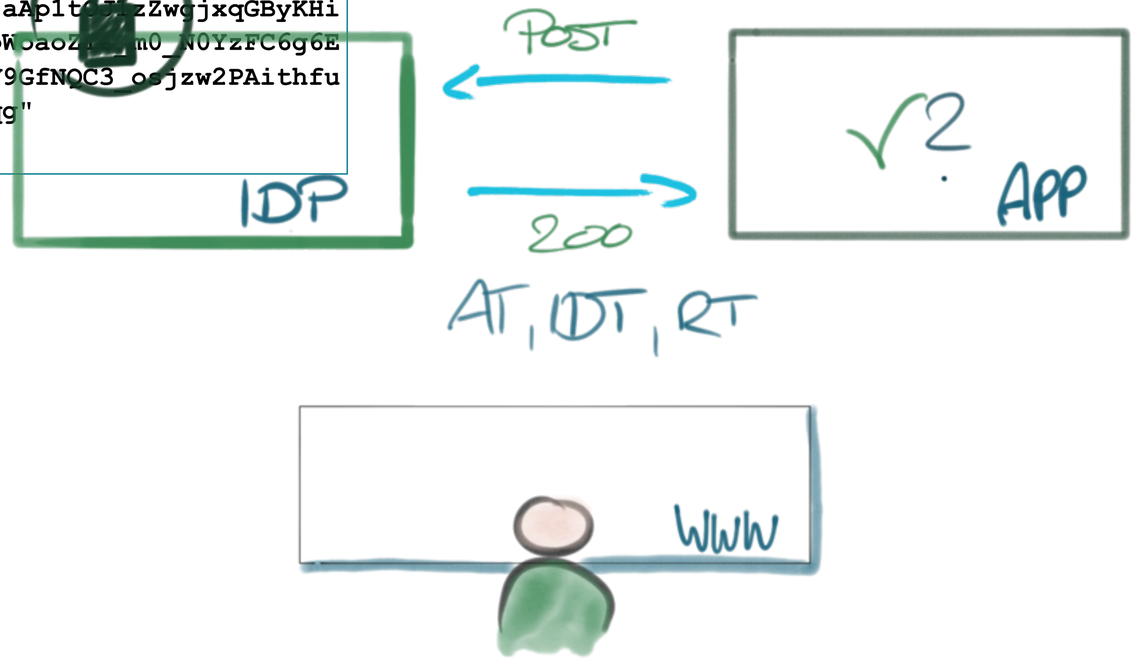
```
HTTP/1.1 302 Found
Location: https://app.de/cb?
  code=SplxlOBeZQQYbYS6WxSbIA
  &state=af0ifjsldkj
```



```
POST /token HTTP/1.1
Host: idp.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fapp.de%2Fcb
```



```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
 "access_token": "SlAV32hkKG",
 "token_type": "Bearer",
 "refresh_token": "8xLOxBtZp8",
 "expires_in": 3600,
 "id_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mj
g5NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0
wUzZfV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEy
ODA5NzAKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU
96Sr_P6qJp6IcmD3HP99Obi1PRscwh3Lp146waJ8IhehcwD7F09JdijmBqkvP
eB2T9CJNqeGpeccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJzZwcjxqGByKHi
OtX7TpQyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZerkm0_N0YzFC6g6E
JbOEoRoSK5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfu
bEEBLuVVkXUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
}
```



Token?

JWT

JSON Web Token

User Info + Metadata

Signed JSON Web Token
Realm Private Key
Limited lifespan
May be revoked

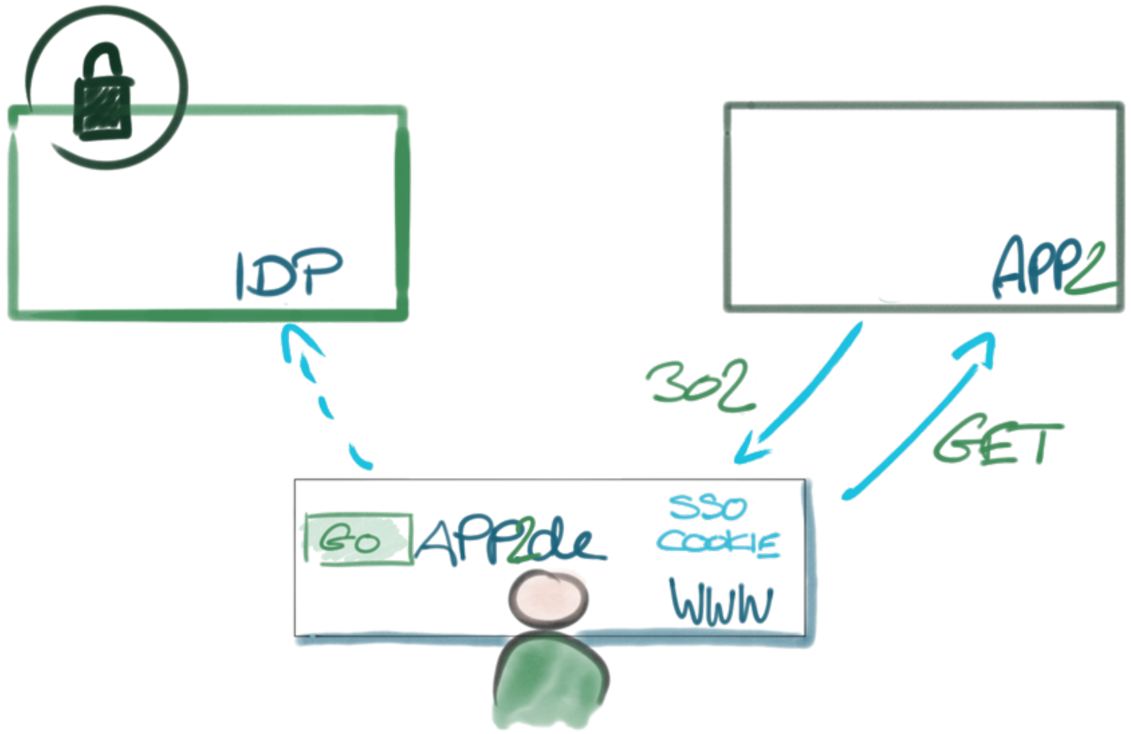
Can be verified

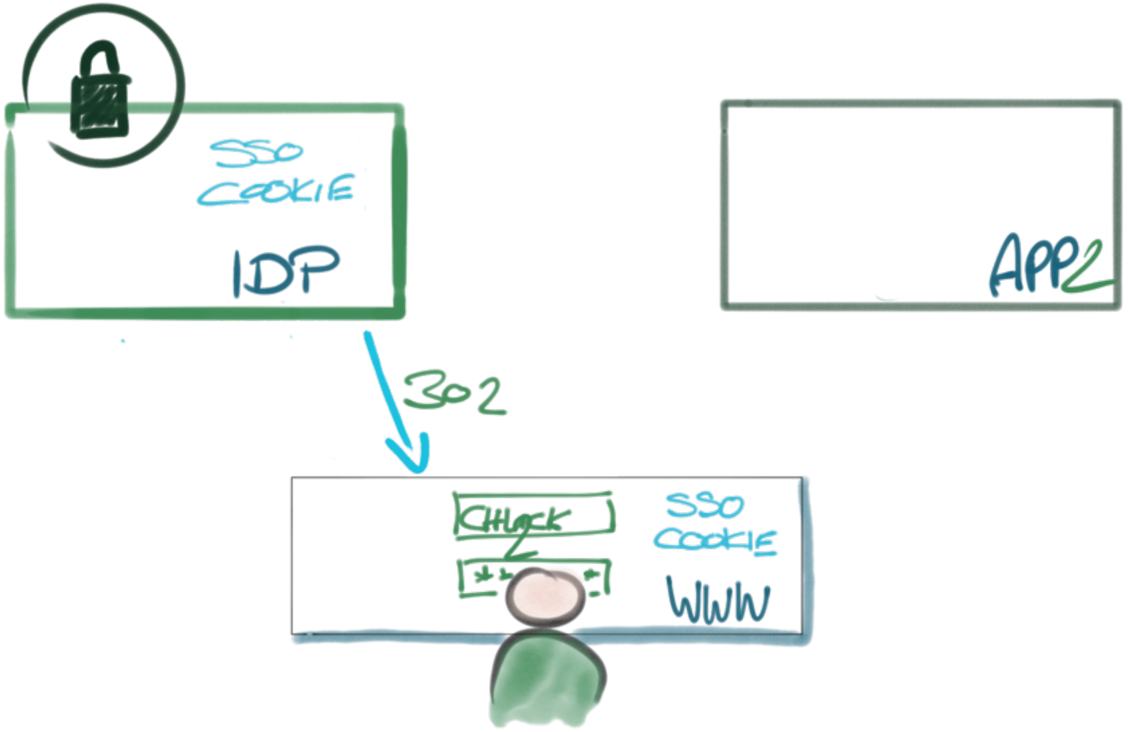
Checking the Signature
POST to token introspection

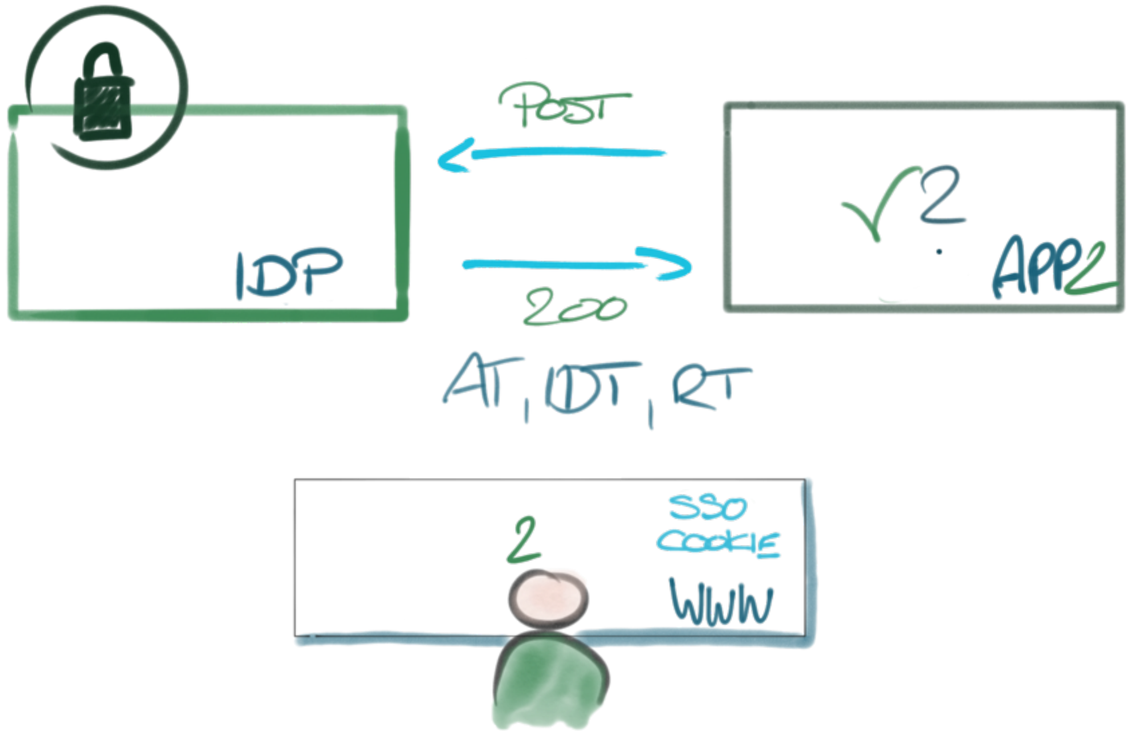
Token Types

Access-Tokens
Refresh-Tokens
Offline-Tokens
Identity-Tokens

Magic **SSO**?

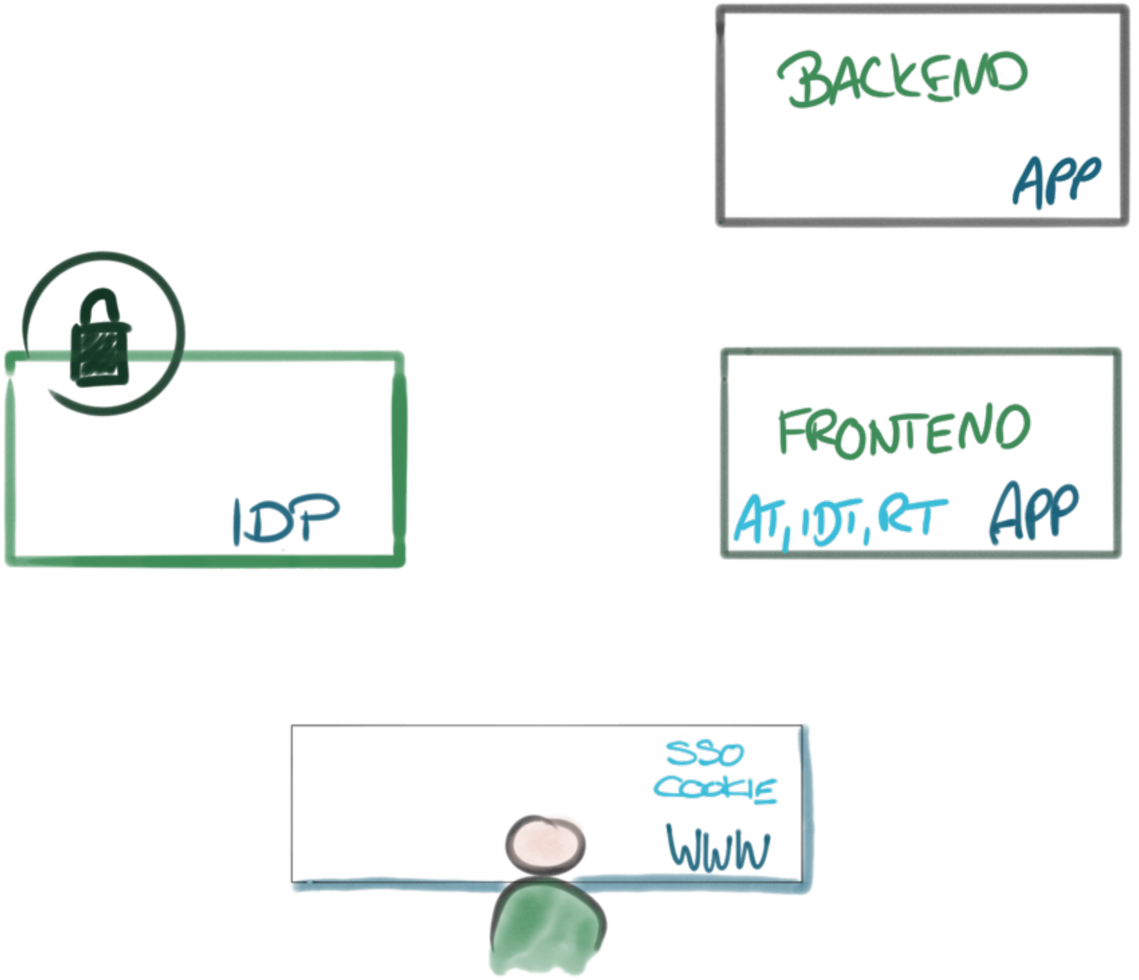

IDP
APP2
302
GET
GO APP2de
SSO COOKIE
WWW


SSO
COOKIE
IDP
APP2
302
XHRack
SSO
COOKIE
WWW

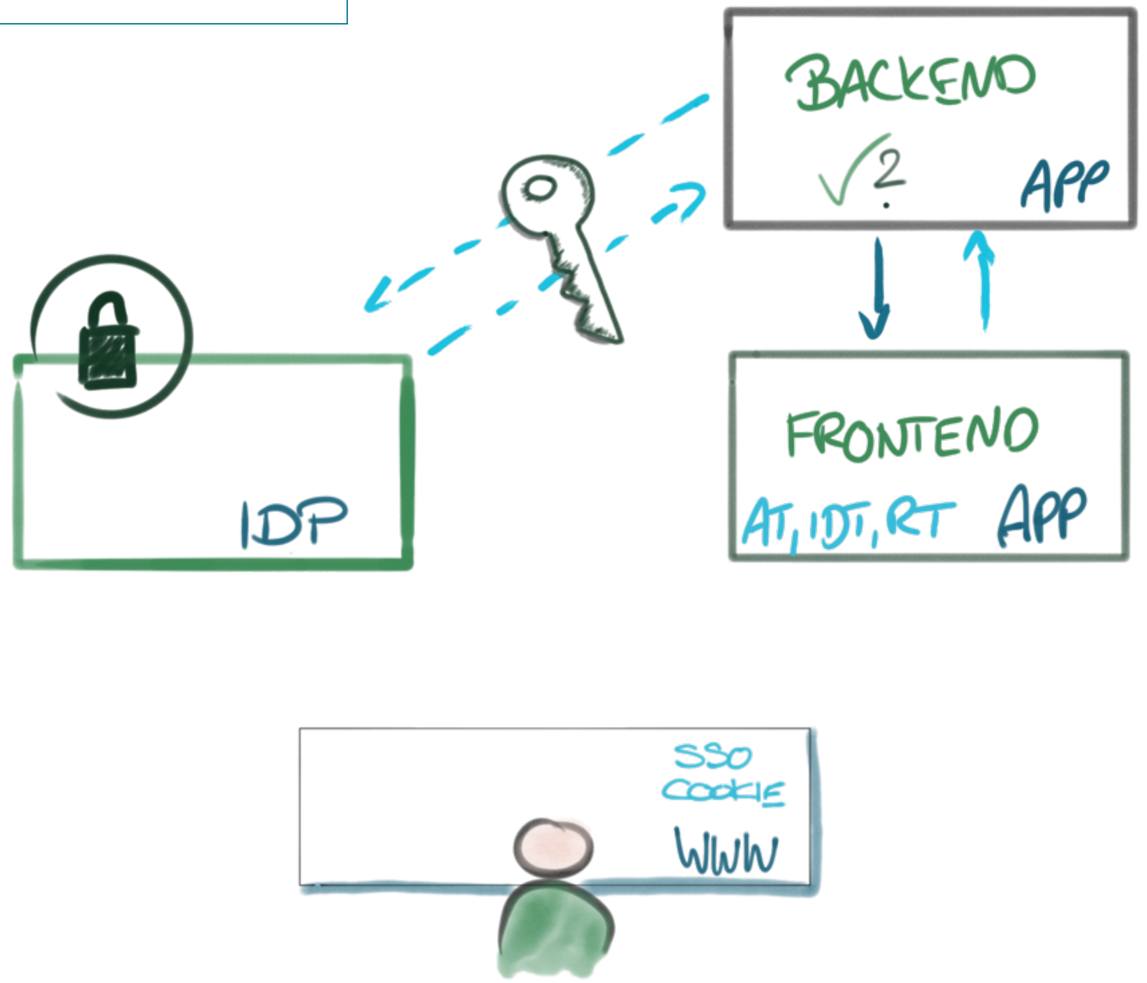

IDP
POST
200
APP2
AT, IDT, RT
2
SSO
COOKIE
WWW

Many more **grants** and **flows**.

How do applications actually **use** token?


BACKEND
APP
IDP
FRONTEND
AT, IDT, RT
APP
SSO
COOKIE
WWW

```
GET /resource/1 HTTP/1.1
    Host: backendapp.com
    Authorization: Bearer mF_9.B5f-4.1JqM
```



Why all the effort?

Centralization of trust.

Here is why a central instance to manage your identity related information is a good idea.

1. Dedicated Login Service, no redundant implementation in every application.
2. Single source of truth and trust.
3. Standardized approach to prevent vendor lock-in.
4. Allow Single Sign-on and Single Logout.

OpenID Connect Provider

Keycloak

An experience report from using this tool more than 2 years in plenty of projects.

Auth0

AWS Cognito

There are very many **other** IDP!

facebook

GitHub

twitter

But we made **very good** experiences when using Keycloak.

Keycloak

The project

Red Hat

Apache Licensed

Documentation

Comprehensive and very many examples.

Open Source

Identity and Access Management

Community

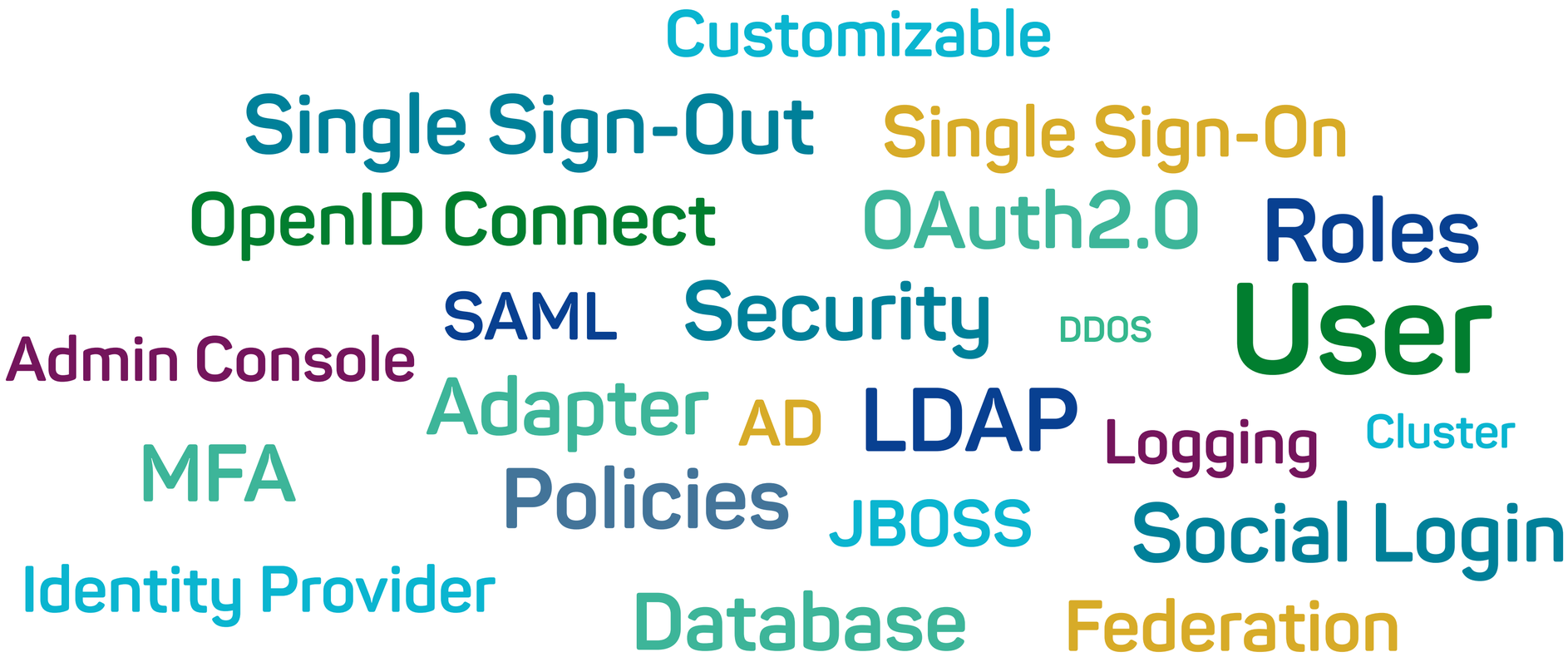
Very vital. Good contact to Developers.

Powerful

Easy to integrate, very many capabilities.

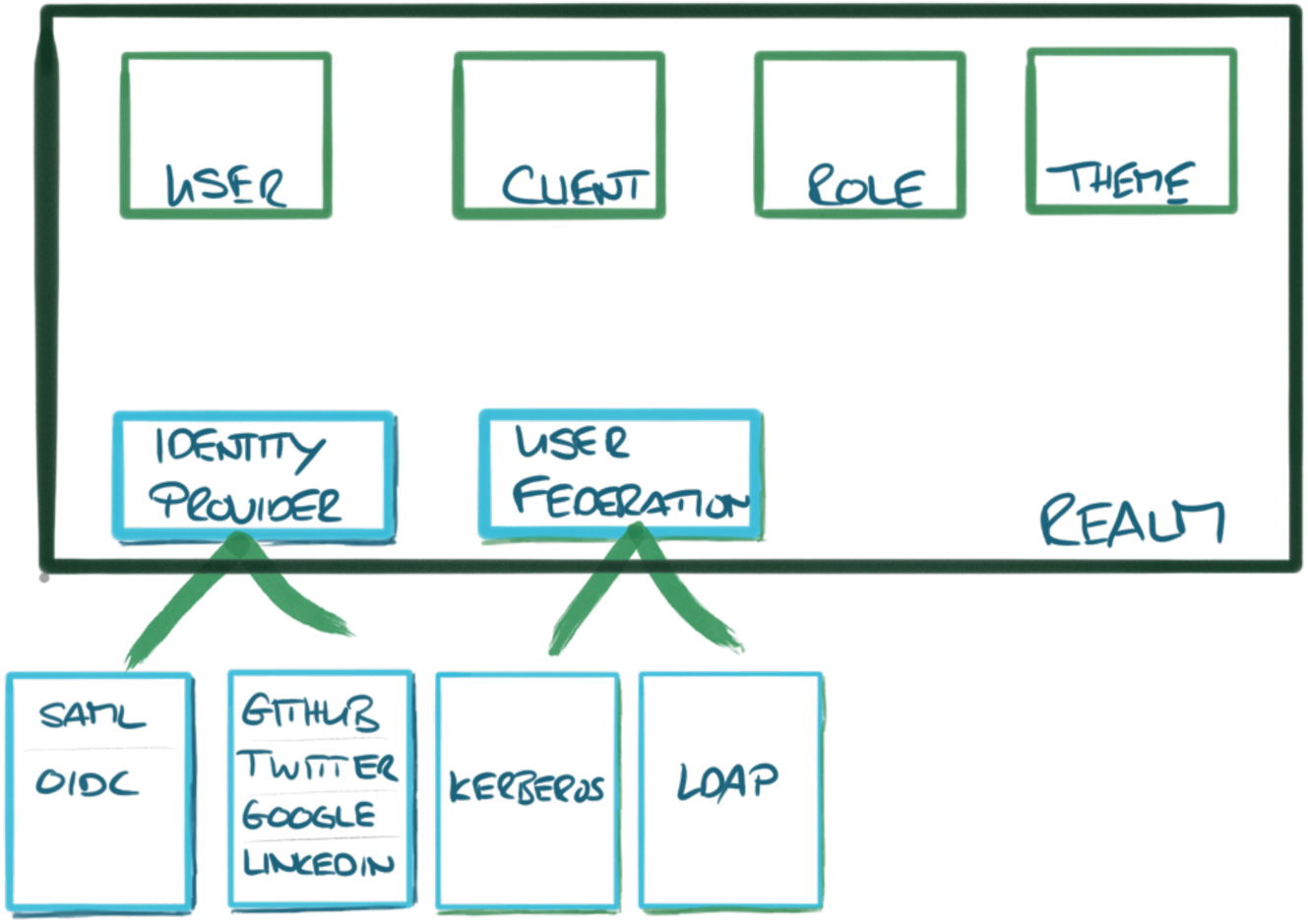
Keycloak

Features



Keycloak

Concepts

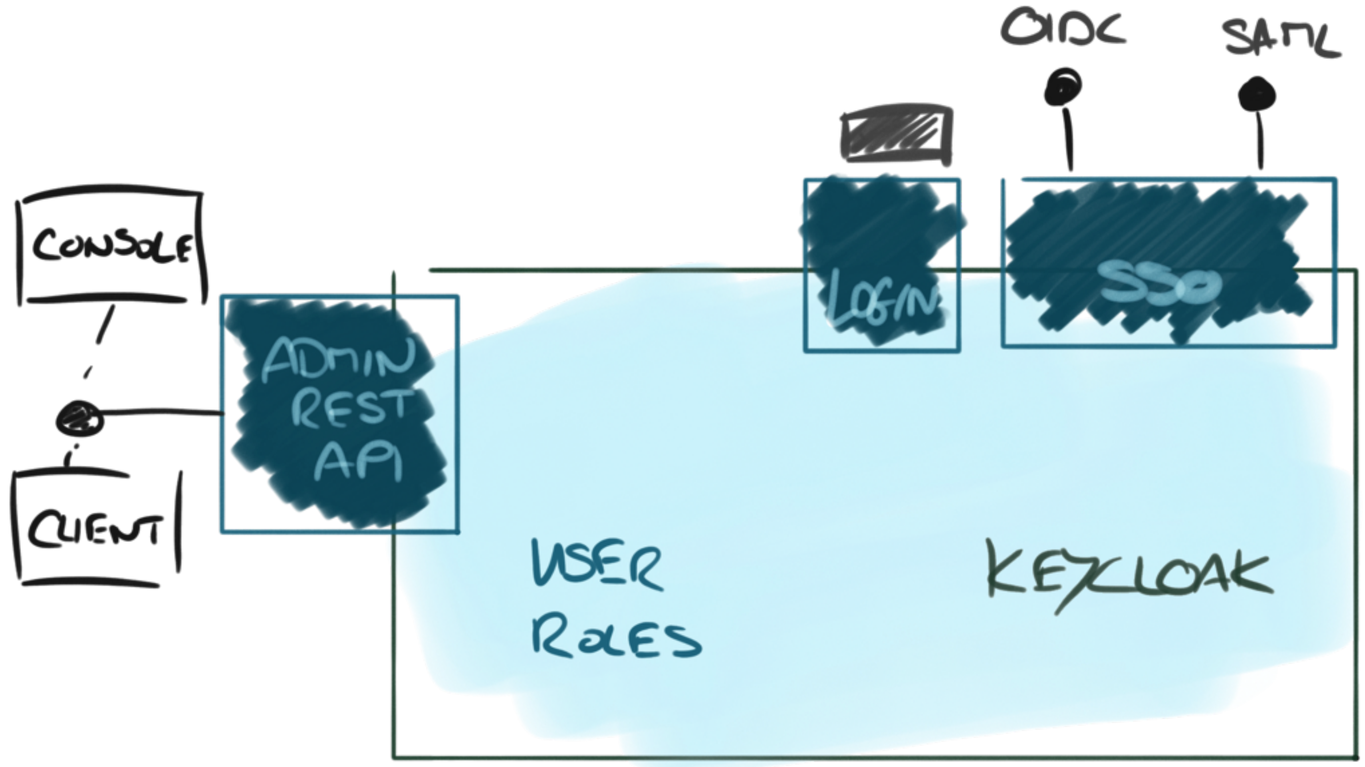


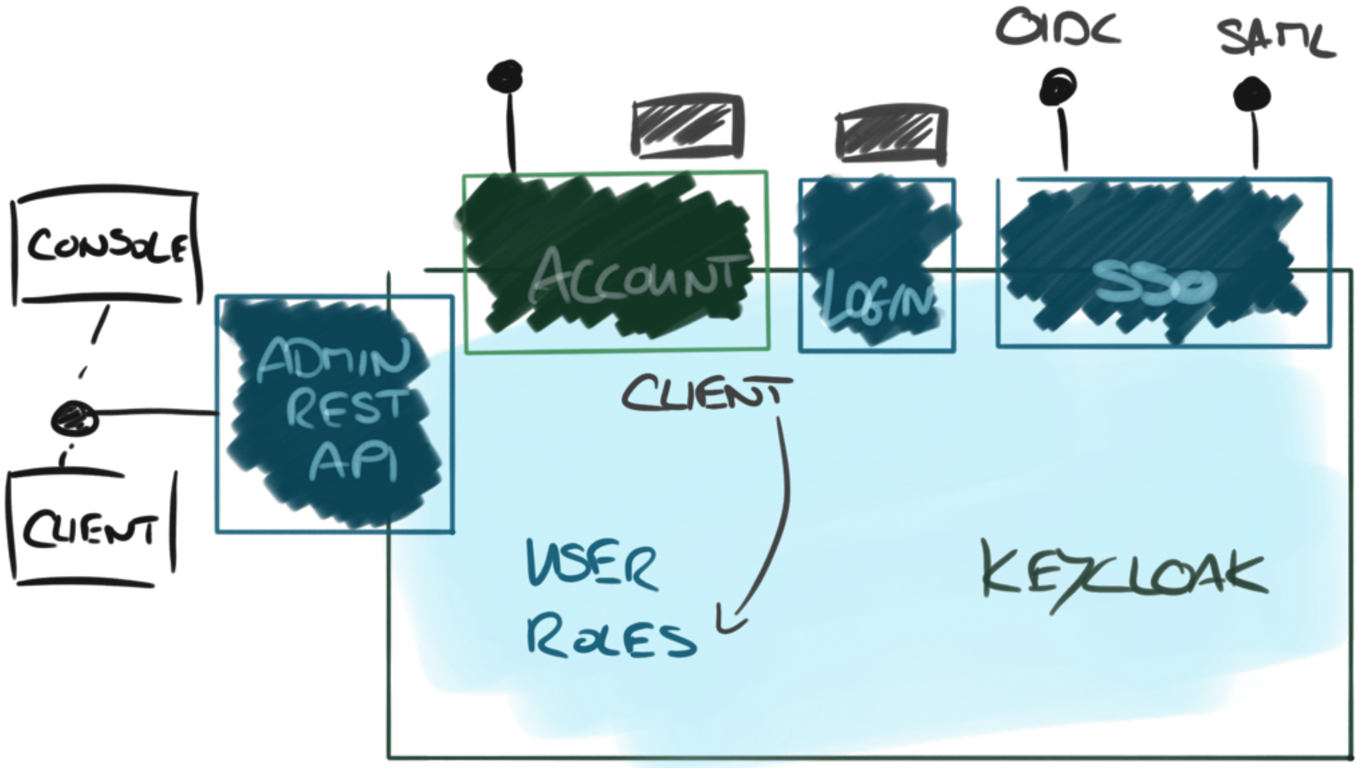
Hands On

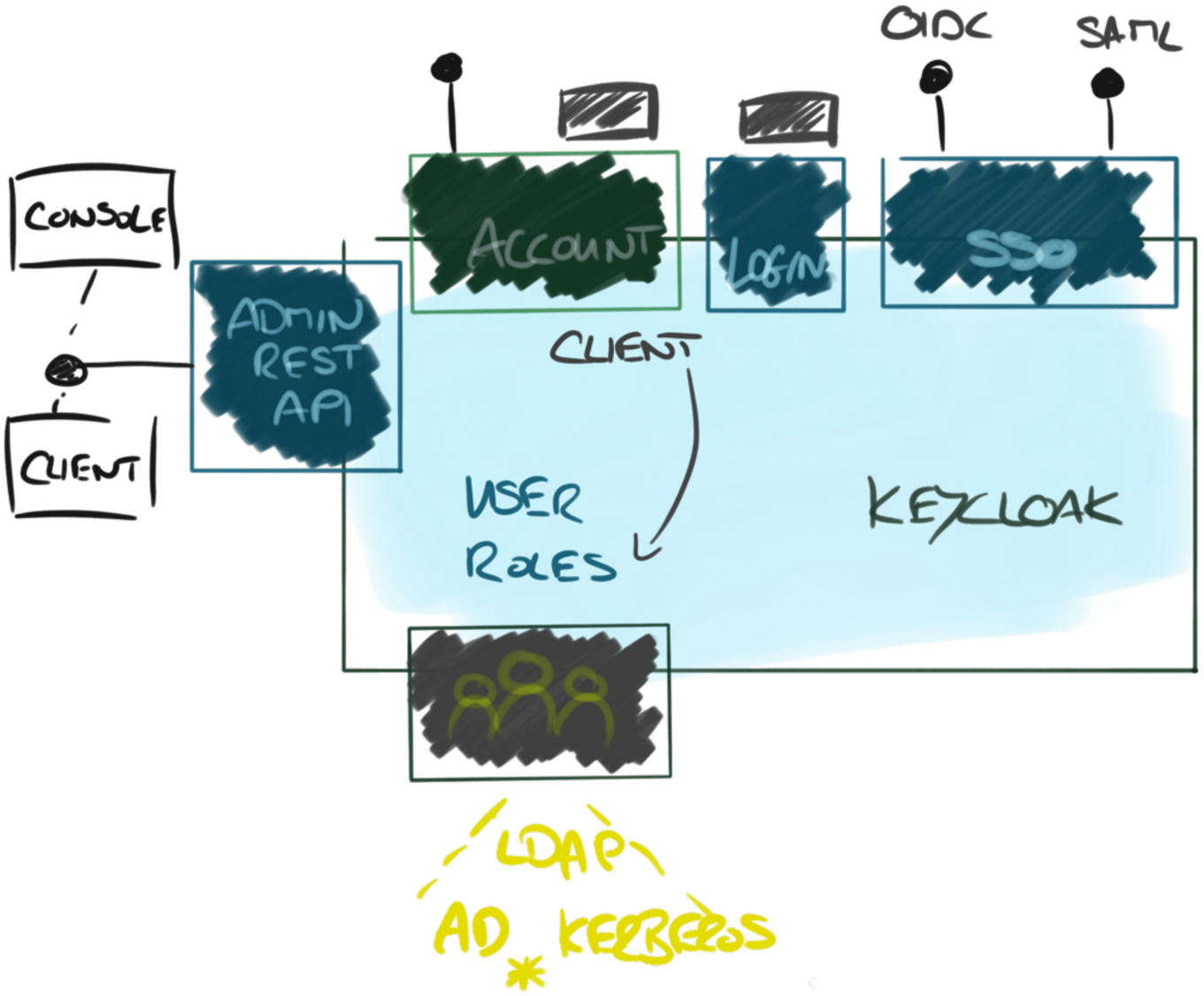
Lets start implementing Keycloak

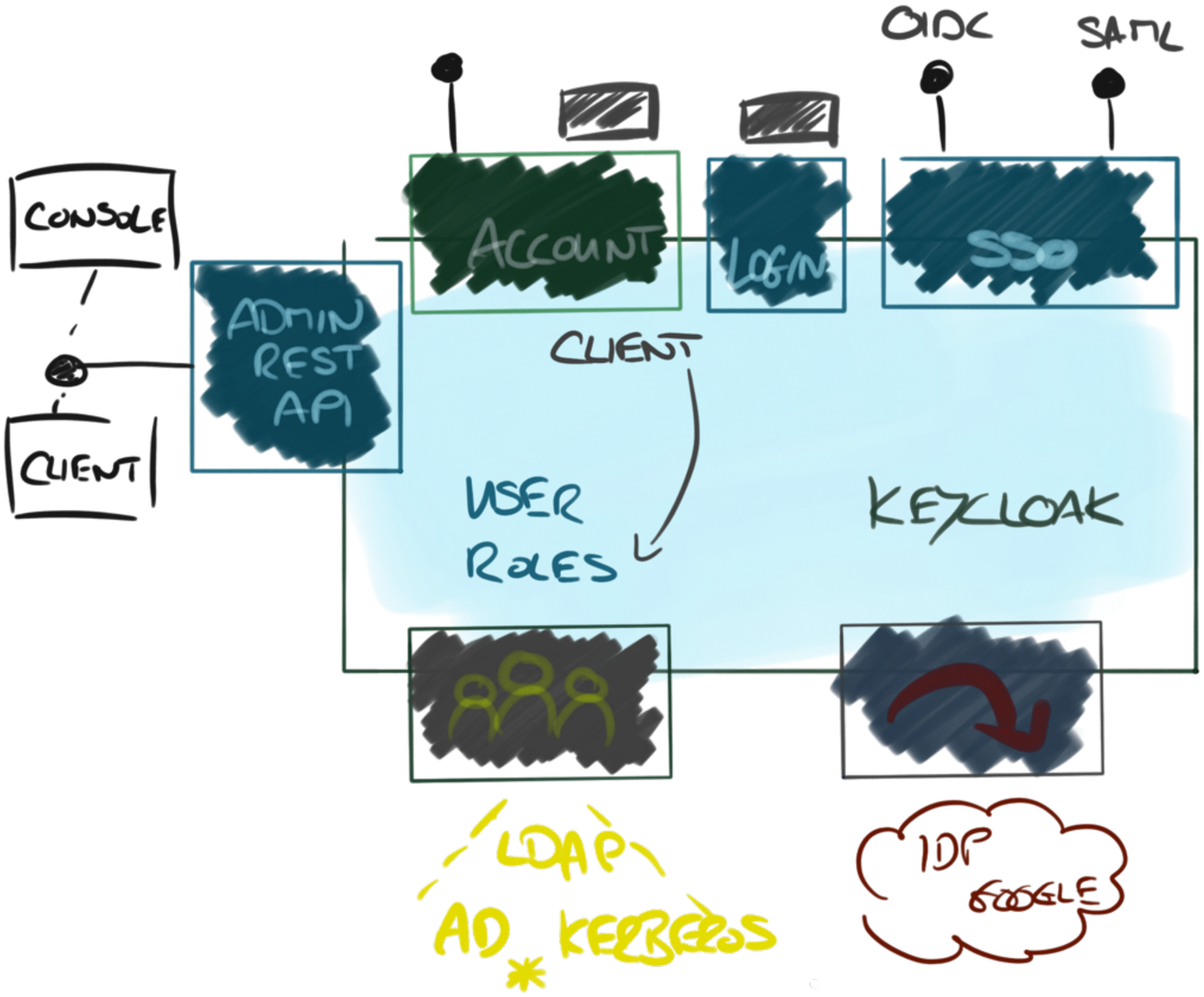
Based on a Use-Case Szenario we now want to show how to integrate Keycloak.

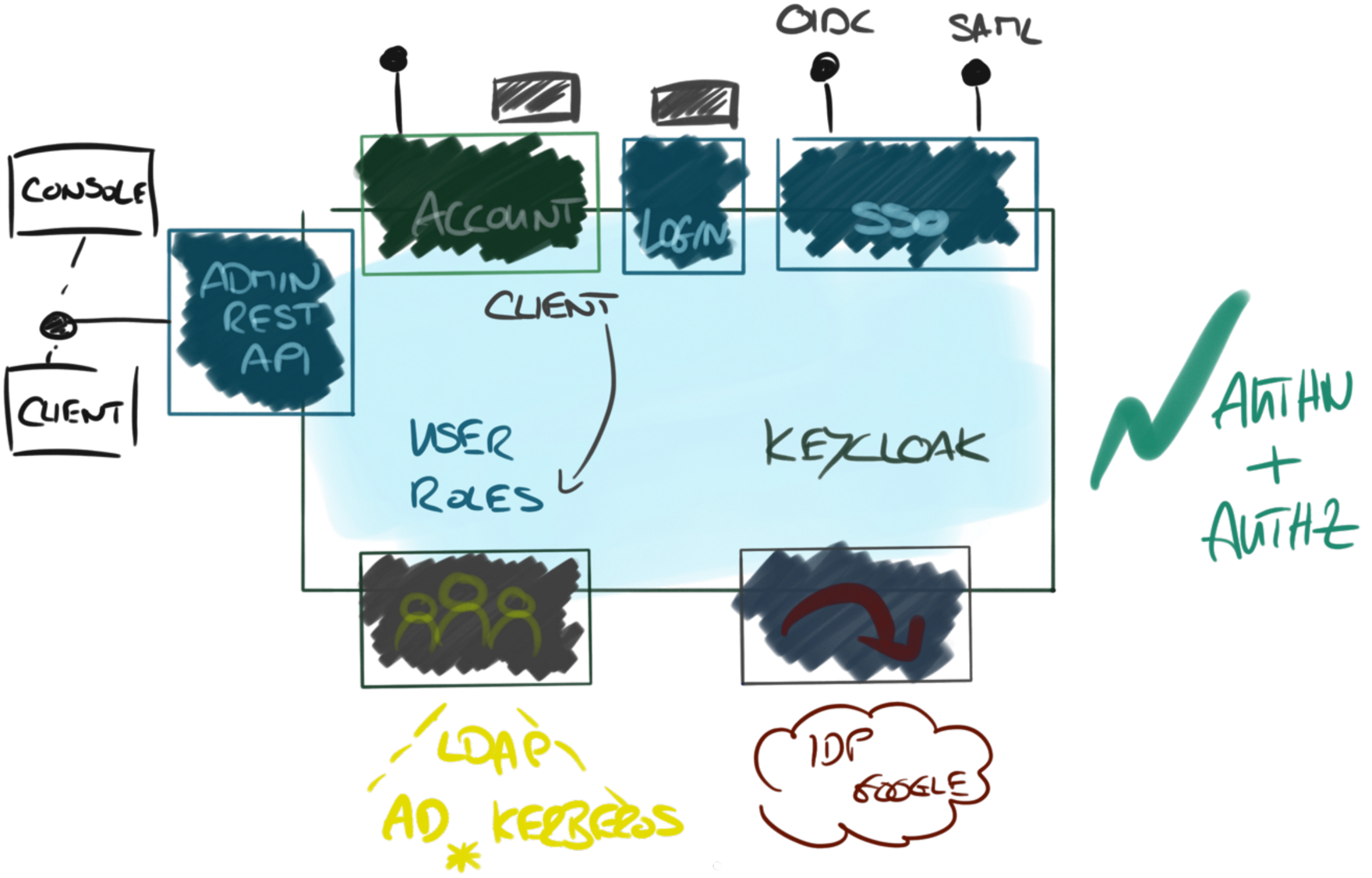

KEYCLOAK

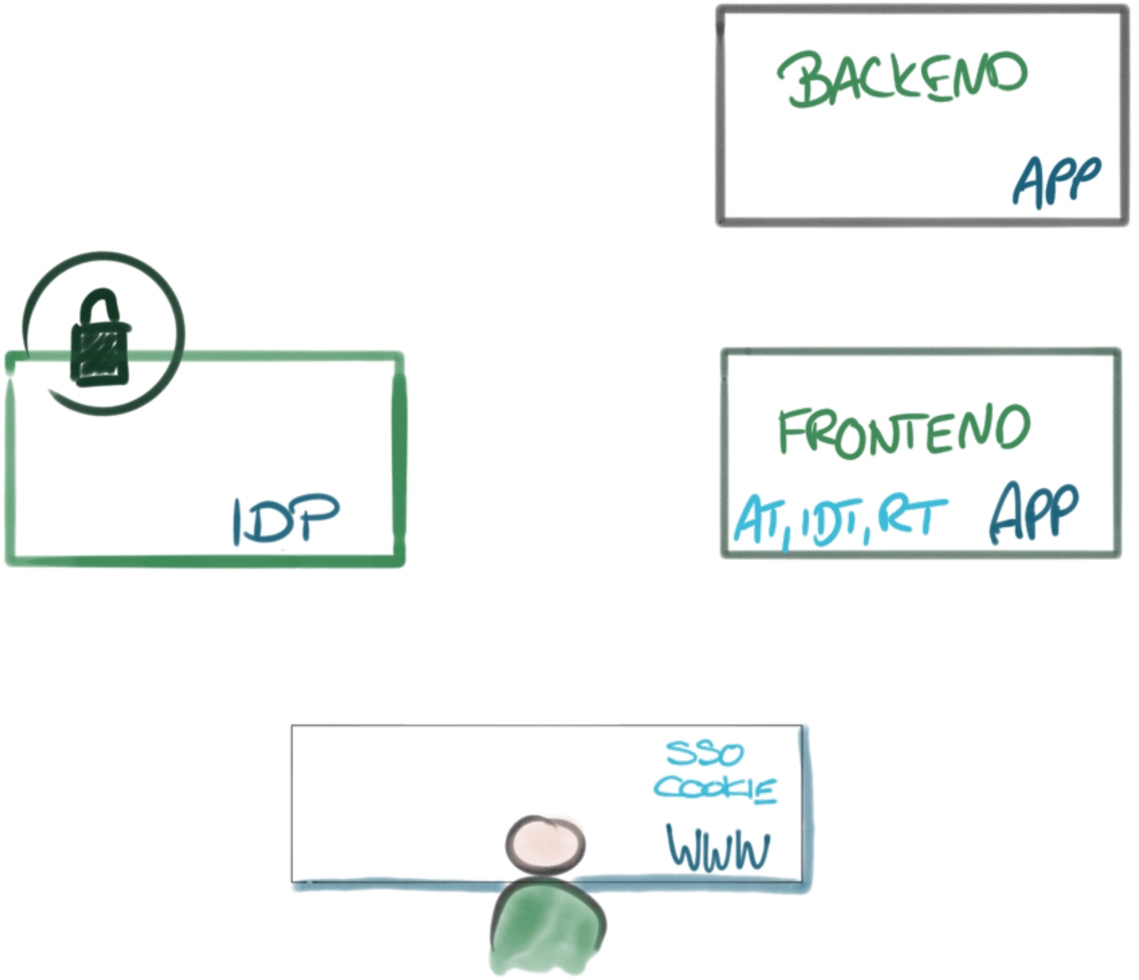

OIDC
SAML
CONSOLE
ADMIN
REST
API
CLIENT
LOGIN
SSO
USER
ROLES
KEYCLOAK


OIDC
SAML
CONSOLE
ACCOUNT
LOGIN
SSO
ADMIN
REST
API
CLIENT
CLIENT
USER
ROLES
KEYCLOAK


OIDC
SAML
CONSOLE
ACCOUNT
LOGIN
SSO
ADMIN REST API
CLIENT
CLIENT
USER ROLES
KEYCLOAK
LDAP
AD * KERBEROS

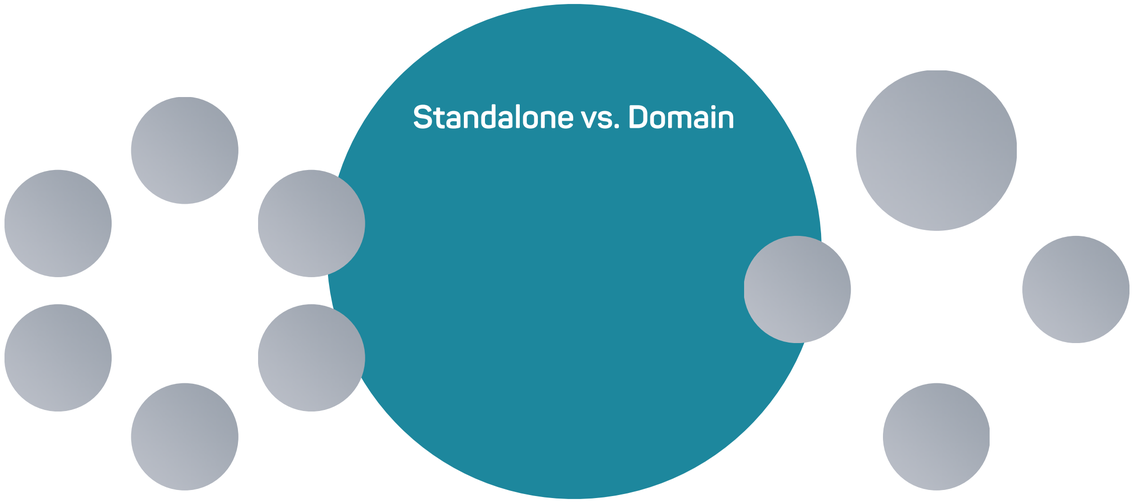

OIDC
SAML
CONSOLE
ACCOUNT
LOGIN
SSO
ADMIN REST API
CLIENT
CLIENT
USER ROLES
KEYCLOAK
LDAP
AD * KERBEROS
IDP
GOOGLE


OIDC
SAML
CONSOLE
ACCOUNT
LOGIN
SSO
ADMIN REST API
CLIENT
CLIENT
USER ROLES
KEYCLOAK
AUTHN + AUTHZ
LDAP
AD * KERBEROS
IDP GOOGLE


BACKEND
APP
IDP
FRONTEND
AT, IDT, RT
APP
SSO
COOKIE
WWW

Operation Mode

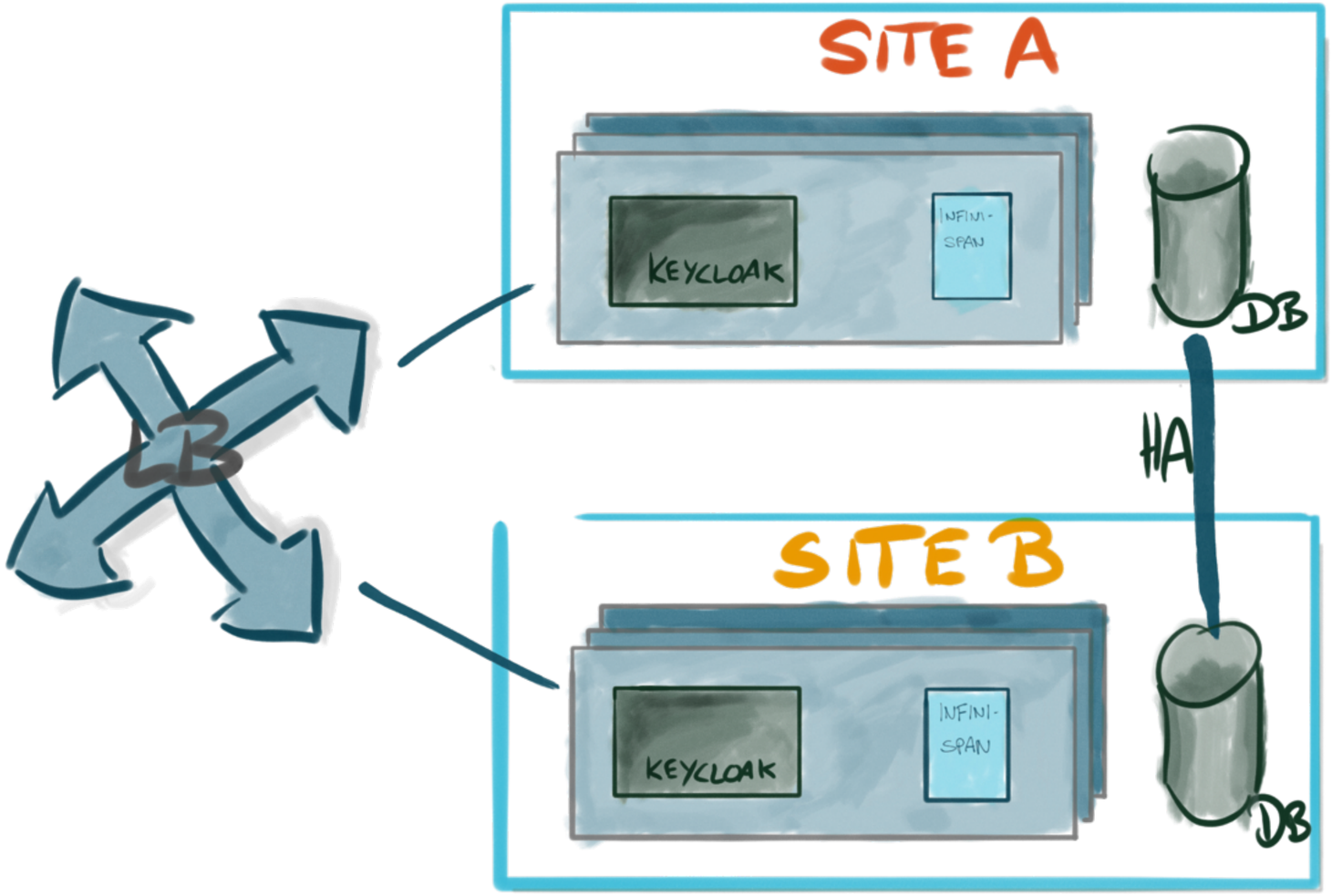
How to run Keycloak?


Standalone vs. Domain

Replication

How to synchronize data?

Caching and persistent storage.



Reliability vs. Performance

Active/Active

Active/Passive

Cache

Authentication Session

authenticationSession

Browser involved, rely on Sticky Session

Action Token Session

actionTokens

e.g. Actions after First Login, Replication

Persistent Data

realms, users, authorization, work

Avoid unnecessary requests to DB, Replication depends

User Session

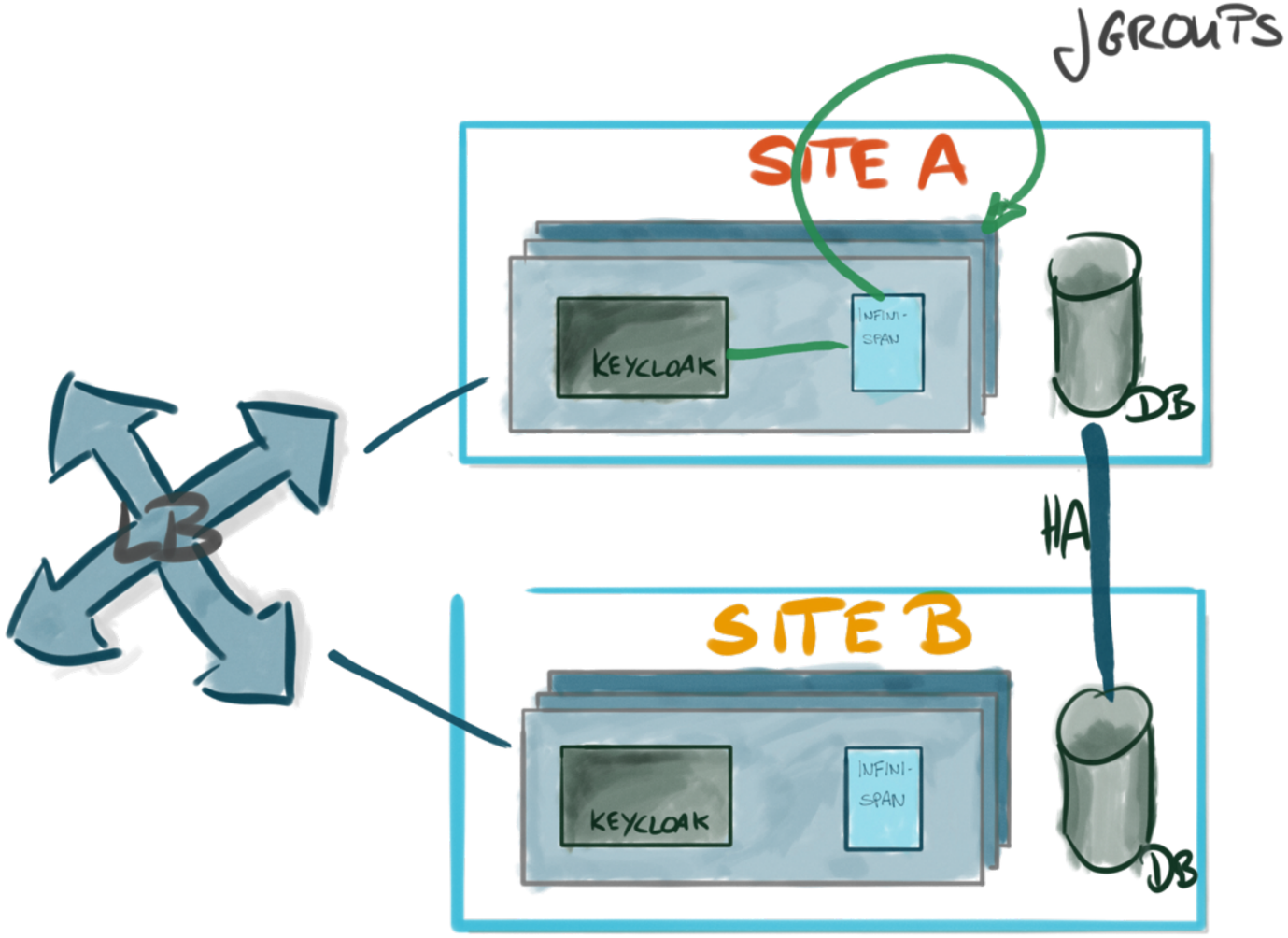
sessions, clientSessions, offlineSessions, offlineClientSessions

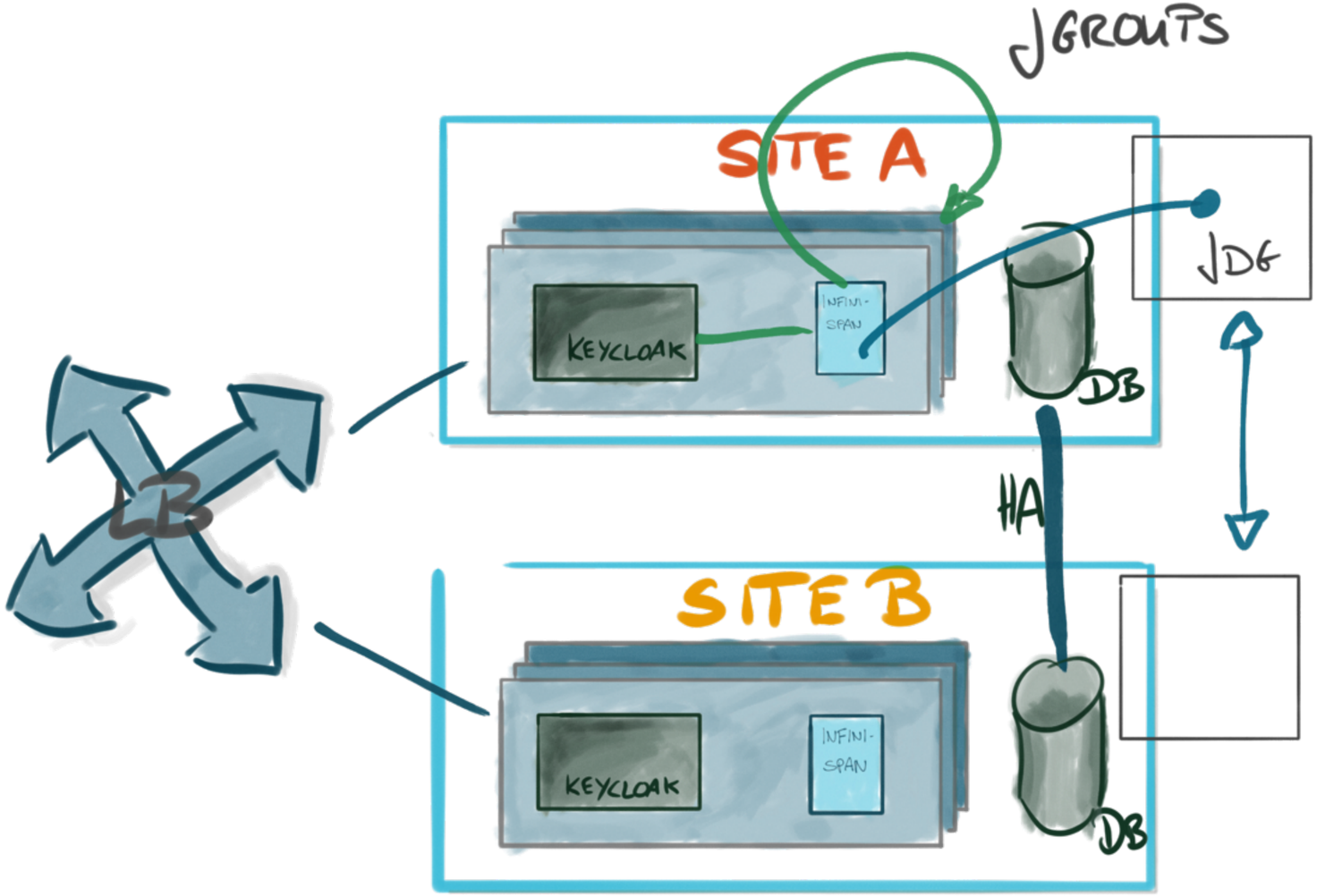
Data of a User Session (browser & app), Replication

Brute Force

loginFailures

Data of a User Session (browser & app), Replication


JGROUPS
SITE A
KEYCLOAK
INFINI-
SPAN
DB
LB
HA
SITE B
KEYCLOAK
INFINI-
SPAN
DB


JGROUPS
SITE A
KEYCLOAK
INFINI-
SPAN
JDG
DB
LB
HA
SITE B
KEYCLOAK
INFINI-
SPAN
DB

Bringing sites offline and online

Scenario: site1 <-> site2 communication is broken

1. Login at site1.
2. site1 tries to write to remote cache.
3. Communication not possible.
4. Exception.
5. No Login possible.

Backup failure policy

FAIL -> WARN, IGNORE

Manually take site offline. (Console)

Automatically. (Configuration)

SYNC vs. ASYNC

SYNC

performance << trust; work, sessions, clientSessions

ASYNC

active/passive; actionTokens

active/passive; javascript; geo location

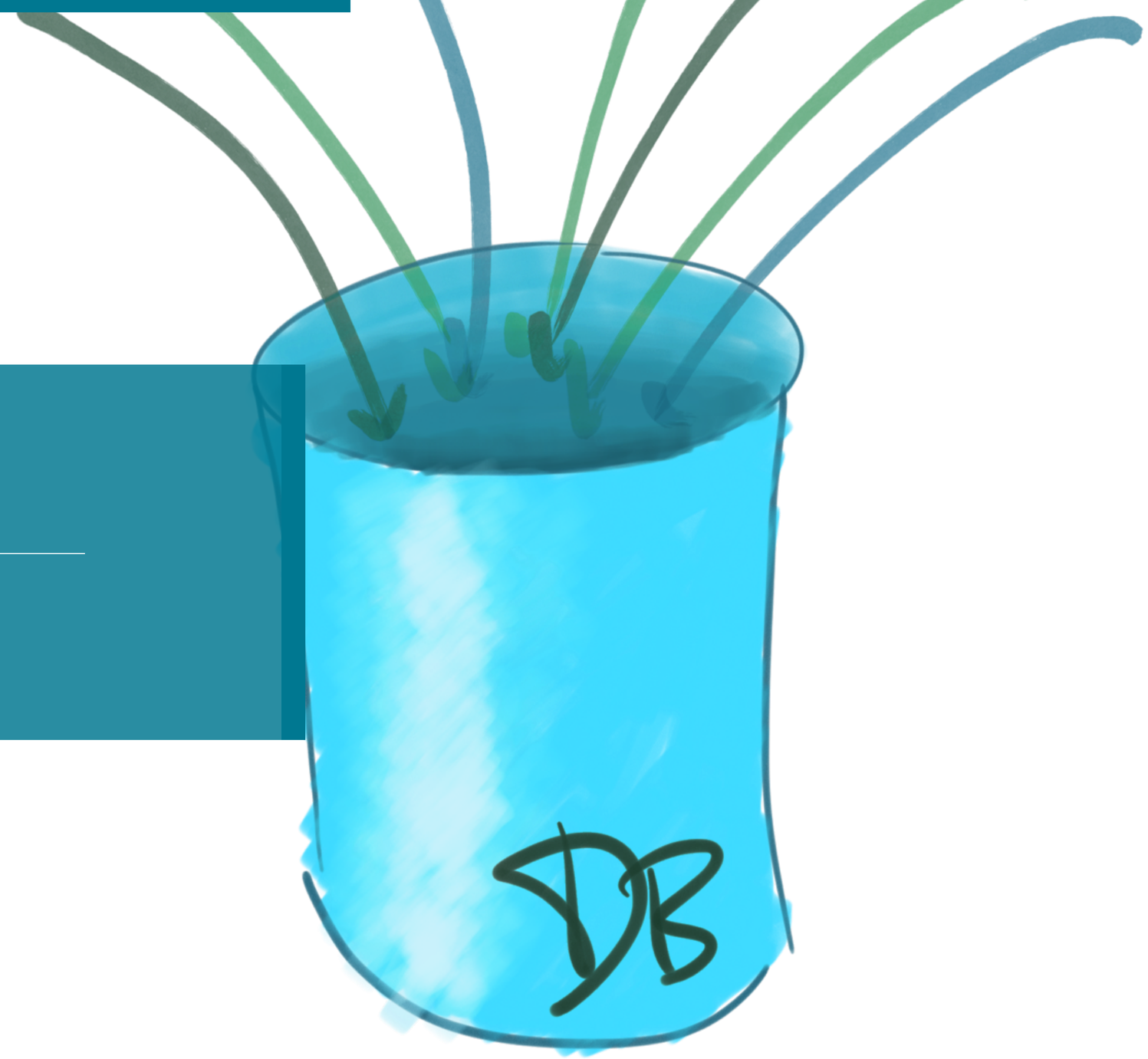
No Backup

loginFailures

Database

Persistent storage

Save it.



Driver

1. JDBC Driver.
2. Driver .jar in module.
3. Declare module in configuration.
4. Modify datasource configuration.
5. Modify connection parameters.

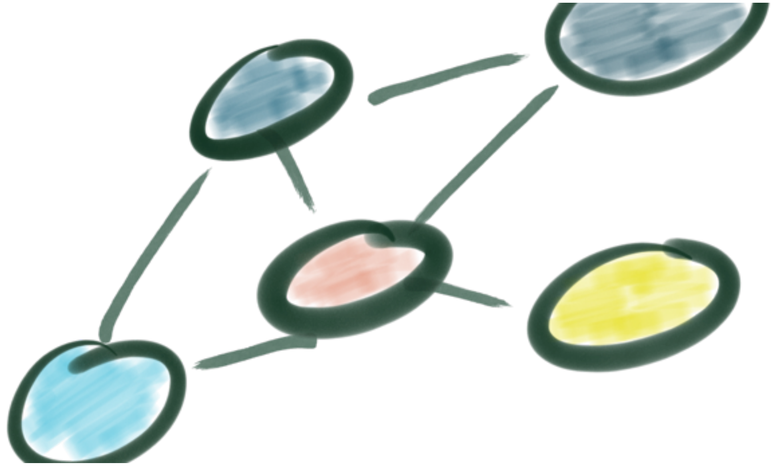

```xml
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.3" name="org.postgresql">

    <resources>
        <resource-root path="postgresql-9.4.1212.jar"/>
    </resources>

    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

```
<subsystem xmlns="urn:jboss:domain:datasources:4.0">
    <datasources>
      ...
      <drivers>
         <driver name="postgresql" module="org.postgresql">
             <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
         </driver>
         <driver name="h2" module="com.h2database.h2">
             <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
         </driver>
      </drivers>
    </datasources>
</subsystem>
```

```
  <subsystem xmlns="urn:jboss:domain:datasources:4.0">
     <datasources>
       ...
       <datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS"
enabled="true" use-java-context="true">
           <connection-url>jdbc:postgresql://localhost/keycloak</connection-url>
           <driver>postgresql</driver>
           <pool>
               <max-pool-size>20</max-pool-size>
           </pool>
           <security>
               <user-name>William</user-name>
               <password>password</password>
           </security>
       </datasource>
        ...
     </datasources>
  </subsystem>
```

Network

A very specific thing.



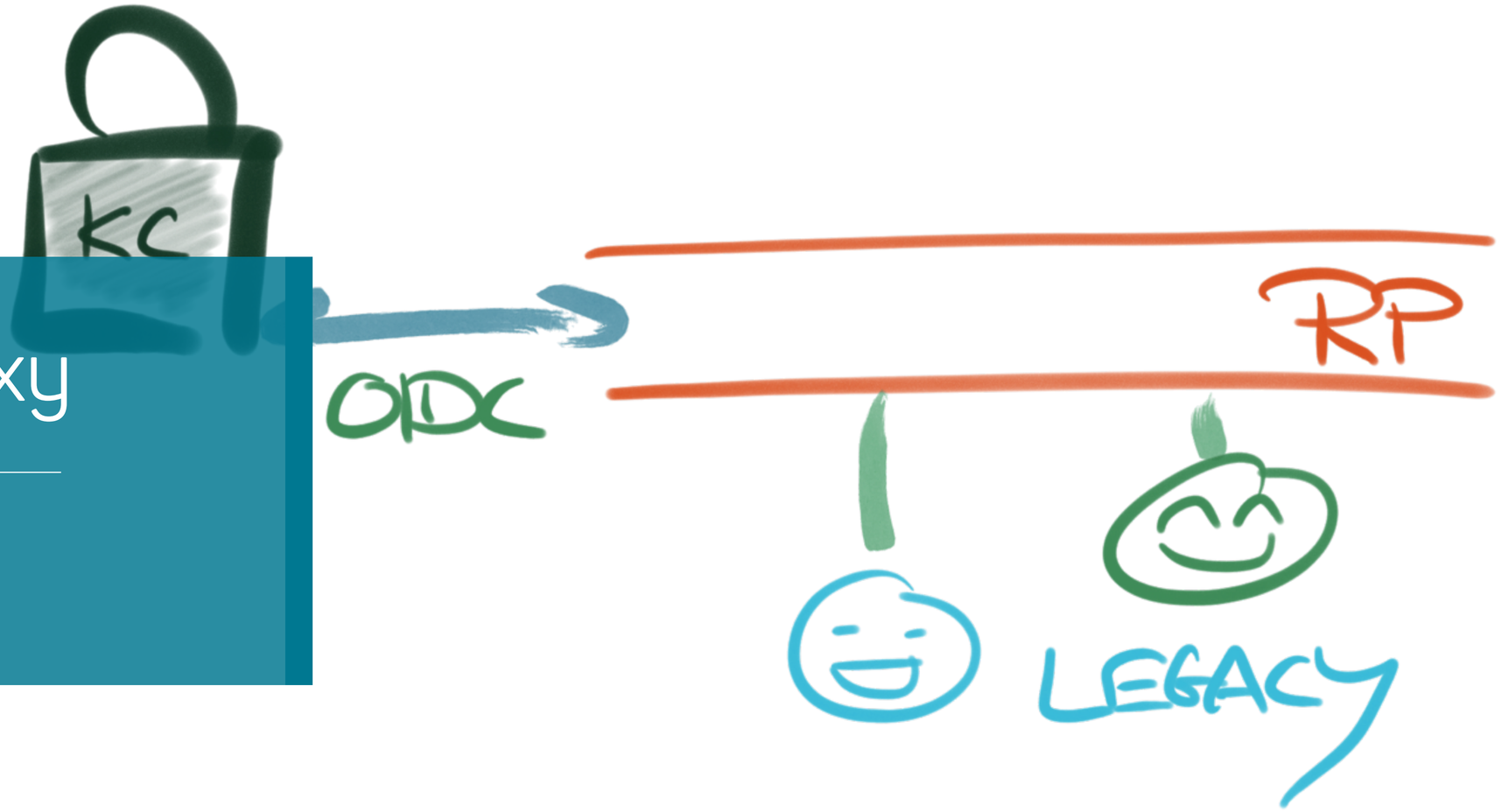
Configure HTTPS/TLS!

```
<security-realm name="UndertowRealm">
    <server-identities>
        <ssl>
            <keystore path="keycloak.jks" relative-to="jboss.server.config.dir" keystore-
password="secret" />
        </ssl>
    </server-identities>
</security-realm>
```

Reverse Proxy

Different approaches

Configure HTTPS/TLS



I'm done!

Address
codecentric AG
Am Mittelhafen 14
58155 Münster

Contact Info
E-Mail: jannik.huels@codecentric.de

Telephone
Telefon: +49 (0) 172.149 0850

@jannikhuels