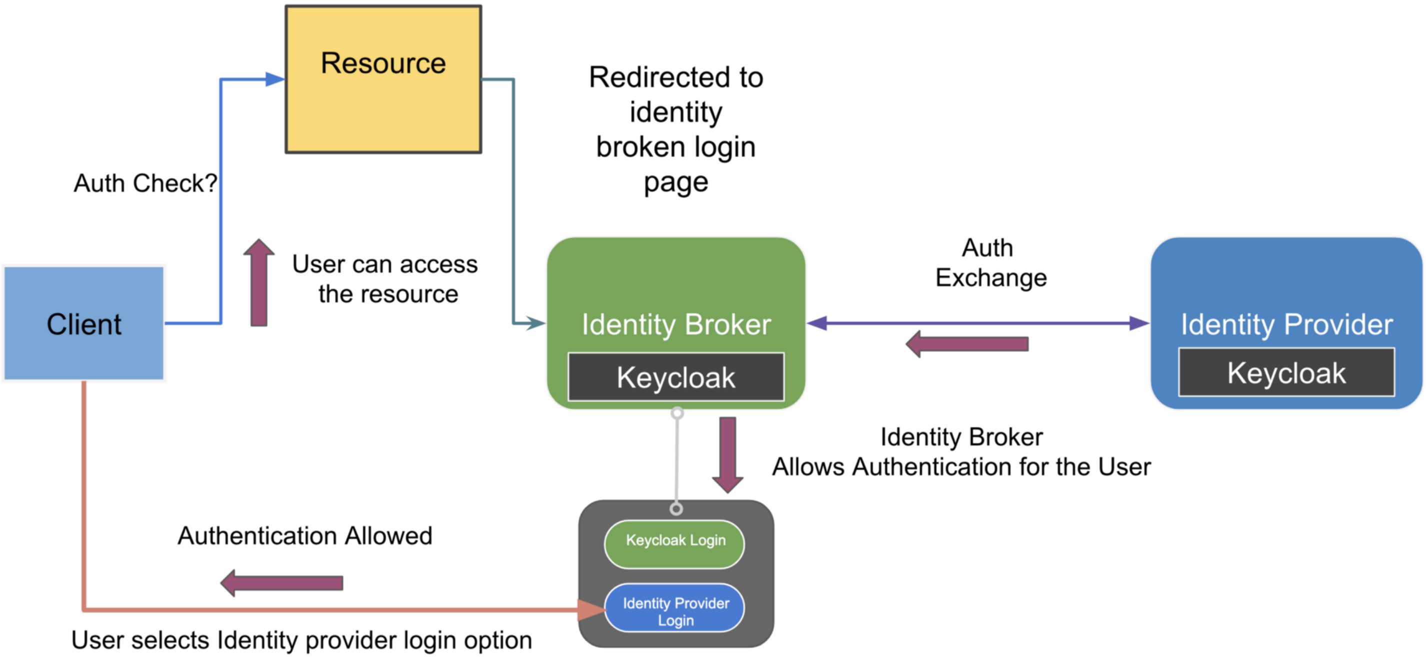
Keycloak as an Identity Broker & an Identity Provider

Abhishek koserwal Follow
Sep 12, 2020 · 3 min read

In this post, we will understand the concept of using Keycloak as an identity Broker & an Identity Provider. Let's go over the basic flow before moving ahead.


Resource
Redirected to identity broken login page
Auth Check?
User can access the resource
Client
Identity Broker
Keycloak
Auth Exchange
Identity Provider
Keycloak
Identity Broker Allows Authentication for the User
Authentication Allowed
Keycloak Login
Identity Provider Login
User selects Identity provider login option

Keycloak as an Identity Broker & as an Identity Provider

- Identity Provider: An application or system that manages identity information. Allow you to create & manage it.

> *Eg:- You can read another post which explains using GitHub as a Social Identity provider: Github as Identity Provider in Keycloak*

- Identity Broker: an intermediatory service that lets you connect with the Identity Providers.

> *Eg:- Broker lets you authenticate or authorize using Identity Provider and let you use the resource linked with Broker. Without re-creating authentication.*

Use Case

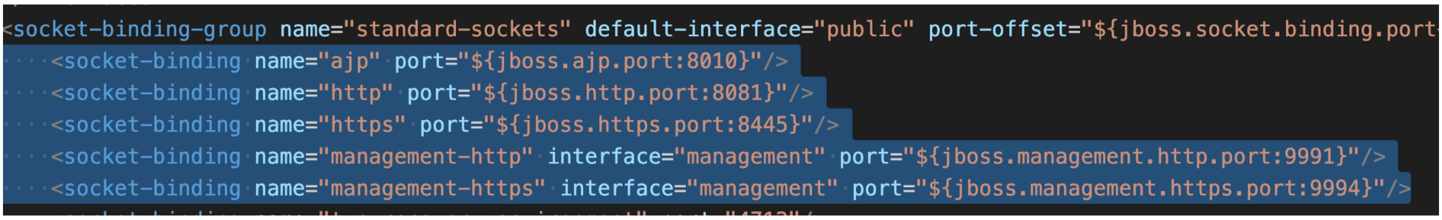
You might have two different Keycloak instance running: one for the external users (Keyloak-External)and another for the internal employees (Keycloak-Internal). Now you want to allow your employees to also authenticate with external service without going for registering a new account the external Keycloak. In such a scenario we can use External Keycloak as Broker and Internal Keycloak as Provider.

Setup

Two Instance of Keycloak Version: 11.0.2

- Keycloak-External (127.0.0.1:8081)
- Keycloak Internal (127.0.0.1:8080) *default*

Update the ports: keycloak-x.x.x/standalone/configuration/standalone.xml



```
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port
    <socket-binding name="ajp" port="${jboss.ajp.port:8010}"/>
    <socket-binding name="http" port="${jboss.http.port:8081}"/>
    <socket-binding name="https" port="${jboss.https.port:8445}"/>
    <socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9991}"/>
    <socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9994}"/>
```

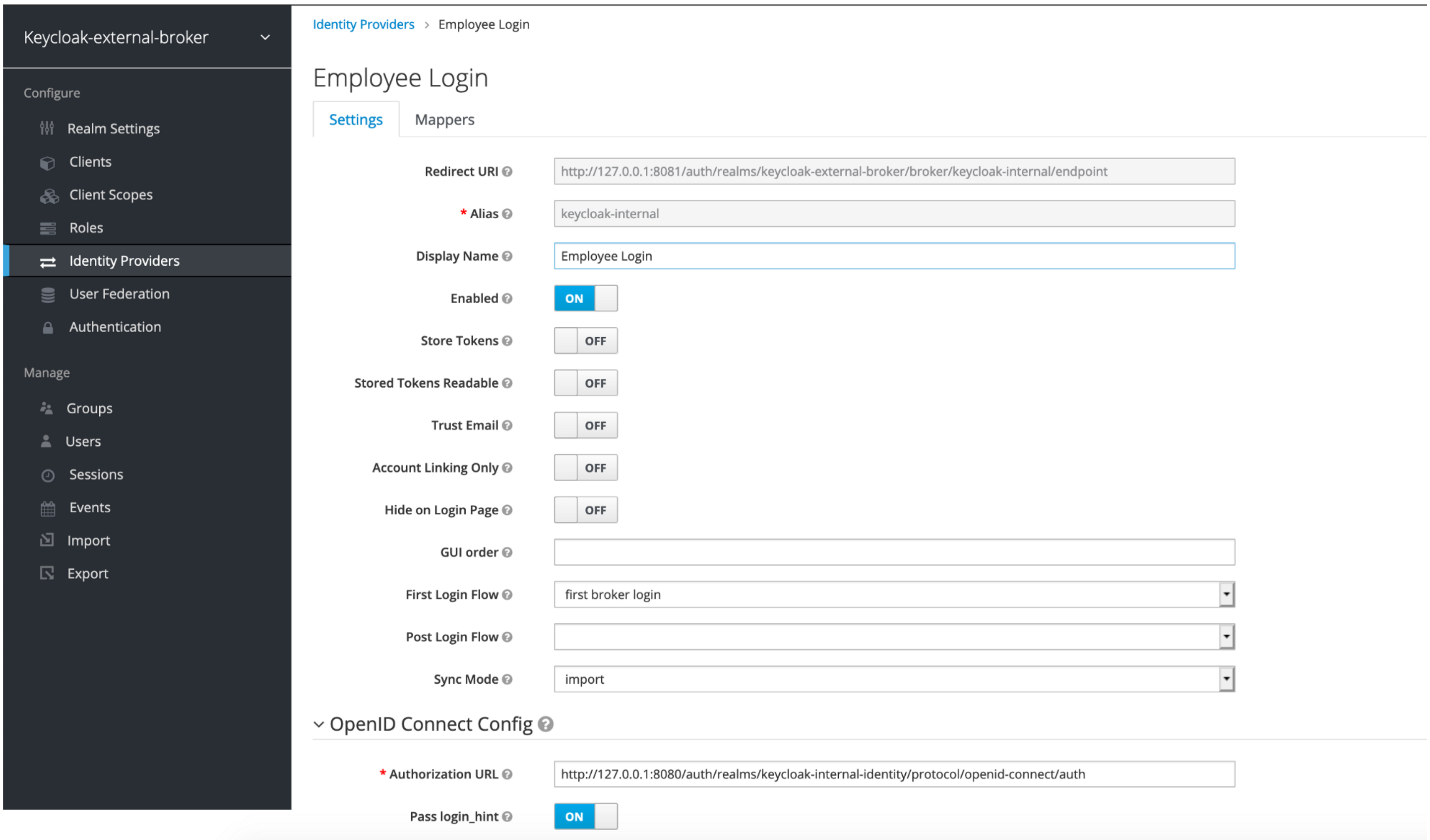
Keycloak External (127.0.0.1:8081)

Create a realm

- realm:"**keycloak-external-broker"** in Keycloak External (127.0.0.1:8081)
- realm **"keycloak-internal-identity"** in Keycloak Internal (127.0.0.1:8080)

Register an Identity Provider

In the Identity Providers: select "Keycloak-odic". I have updated the Alias & Display Name as per our use-case.

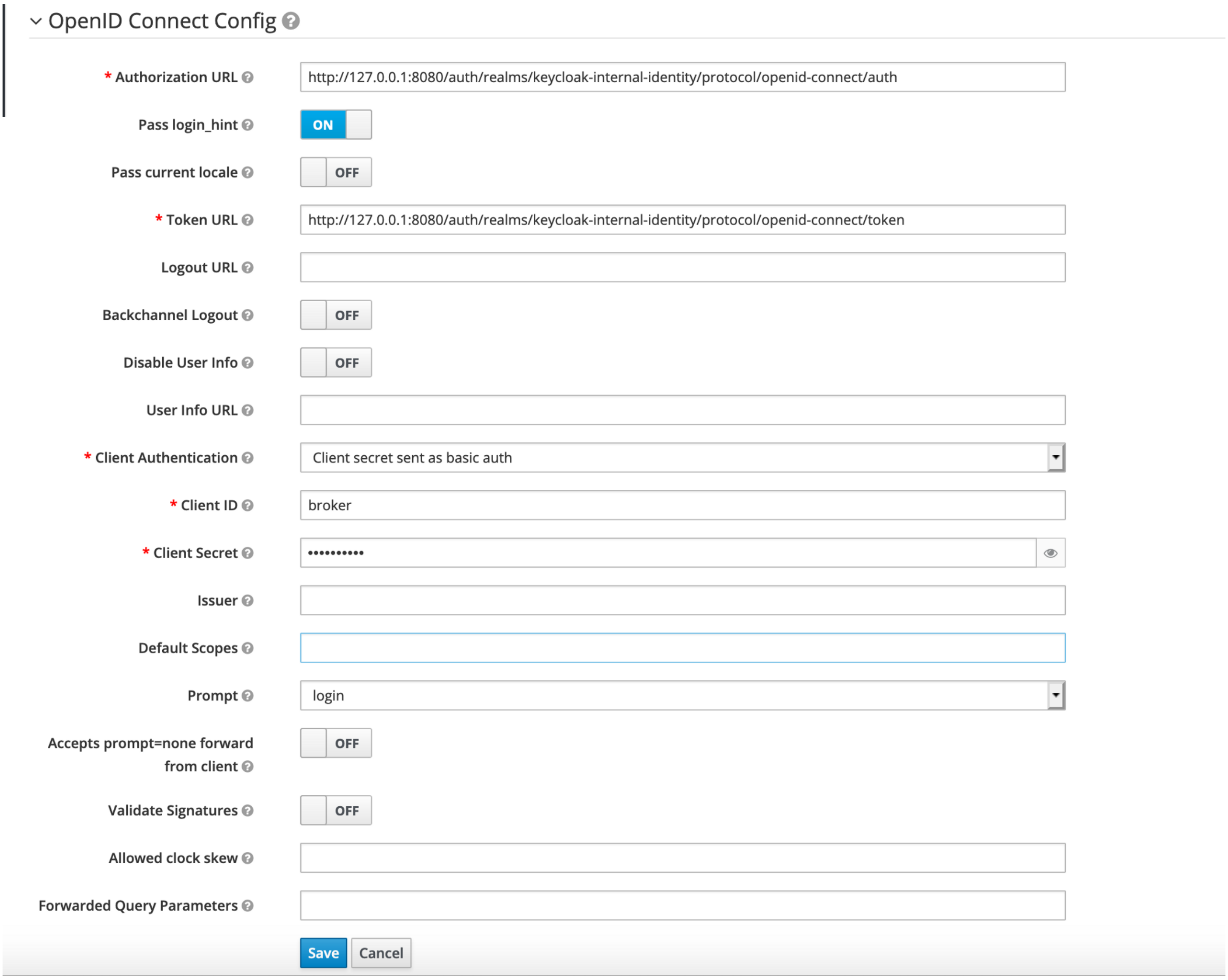


Employee Login

Redirect URI: http://127.0.0.1:8081/auth/realms/keycloak-external-broker/broker/keycloak-internal/endpoint
* Alias: keycloak-internal
Display Name: Employee Login
Enabled: ON
Store Tokens: OFF
Stored Tokens Readable: OFF
Trust Email: OFF
Account Linking Only: OFF
Hide on Login Page: OFF
GUI order:
First Login Flow: first broker login
Post Login Flow:
Sync Mode: import

OpenID Connect Config

* Authorization URL: http://127.0.0.1:8080/auth/realms/keycloak-internal-identity/protocol/openid-connect/auth
Pass login_hint: ON



OpenID Connect Config

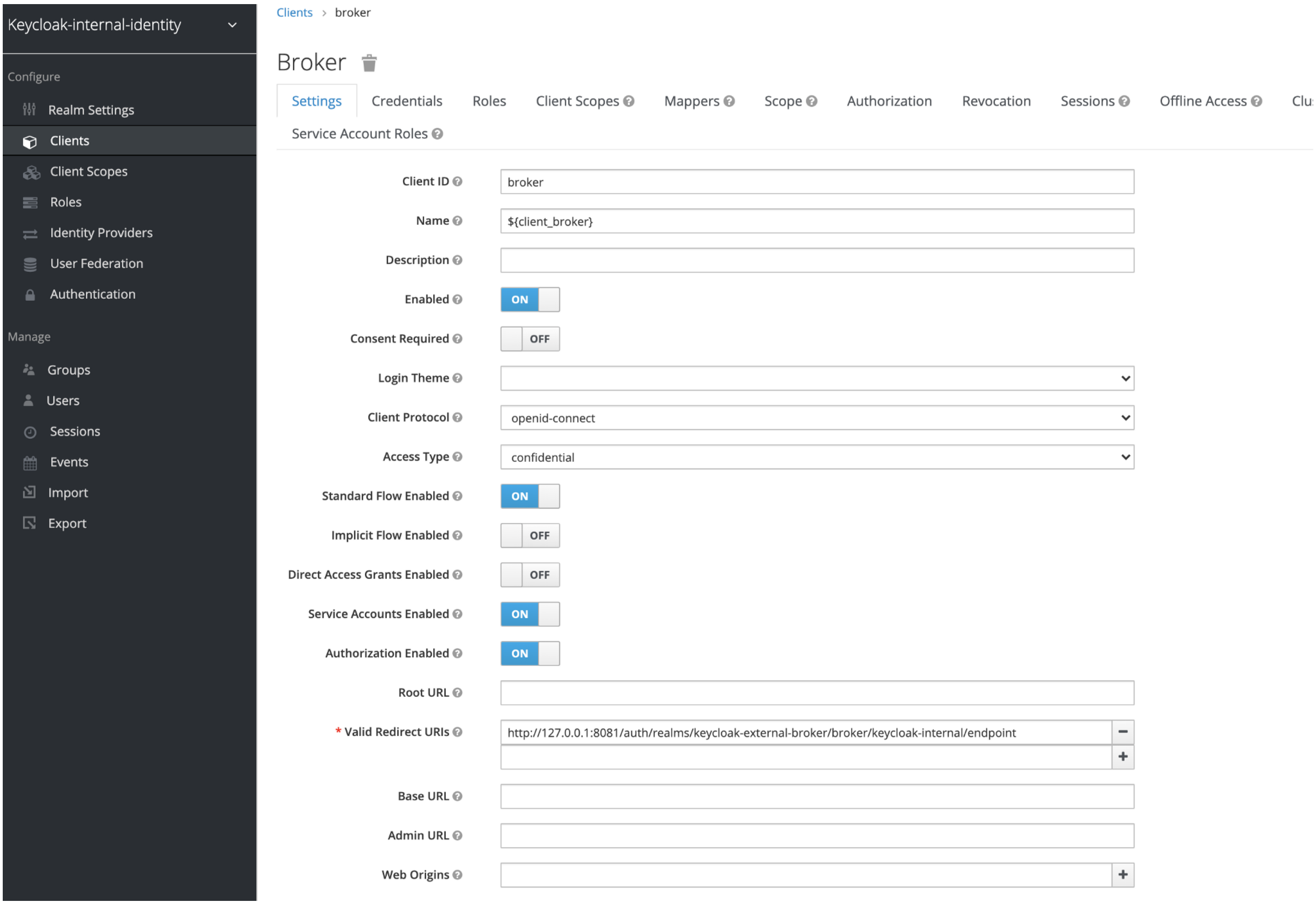
* Authorization URL: http://127.0.0.1:8080/auth/realms/keycloak-internal-identity/protocol/openid-connect/auth
Pass login_hint: ON
Pass current locale: OFF
* Token URL: http://127.0.0.1:8080/auth/realms/keycloak-internal-identity/protocol/openid-connect/token
Logout URL:
Backchannel Logout: OFF
Disable User Info: OFF
User Info URL:
* Client Authentication: Client secret sent as basic auth
* Client ID: broker
* Client Secret: ••••••••••
Issuer:
Default Scopes:
Prompt: login
Accepts prompt=none forward from client: OFF
Validate Signatures: OFF
Allowed clock skew:
Forwarded Query Parameters:

- Configure the Authorization URL: Keycloak internal

```
http://<host>:<ip>/auth/realms/<realm>/protocol/openid-connect/auth
```

- Client ID: Broker
- Client Secret: Copy from the Broker client. (Keycloak Internal: 127.0.0.1:8080)

Keycloak Internal



Client ID	broker
Name	${client_broker}
Description	
Enabled	ON
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	confidential
Standard Flow Enabled	ON
Implicit Flow Enabled	OFF
Direct Access Grants Enabled	OFF
Service Accounts Enabled	ON
Authorization Enabled	ON
Root URL	
* Valid Redirect URIs	http://127.0.0.1:8081/auth/realms/keycloak-external-broker/broker/keycloak-internal/endpoint
Base URL	
Admin URL	
Web Origins	

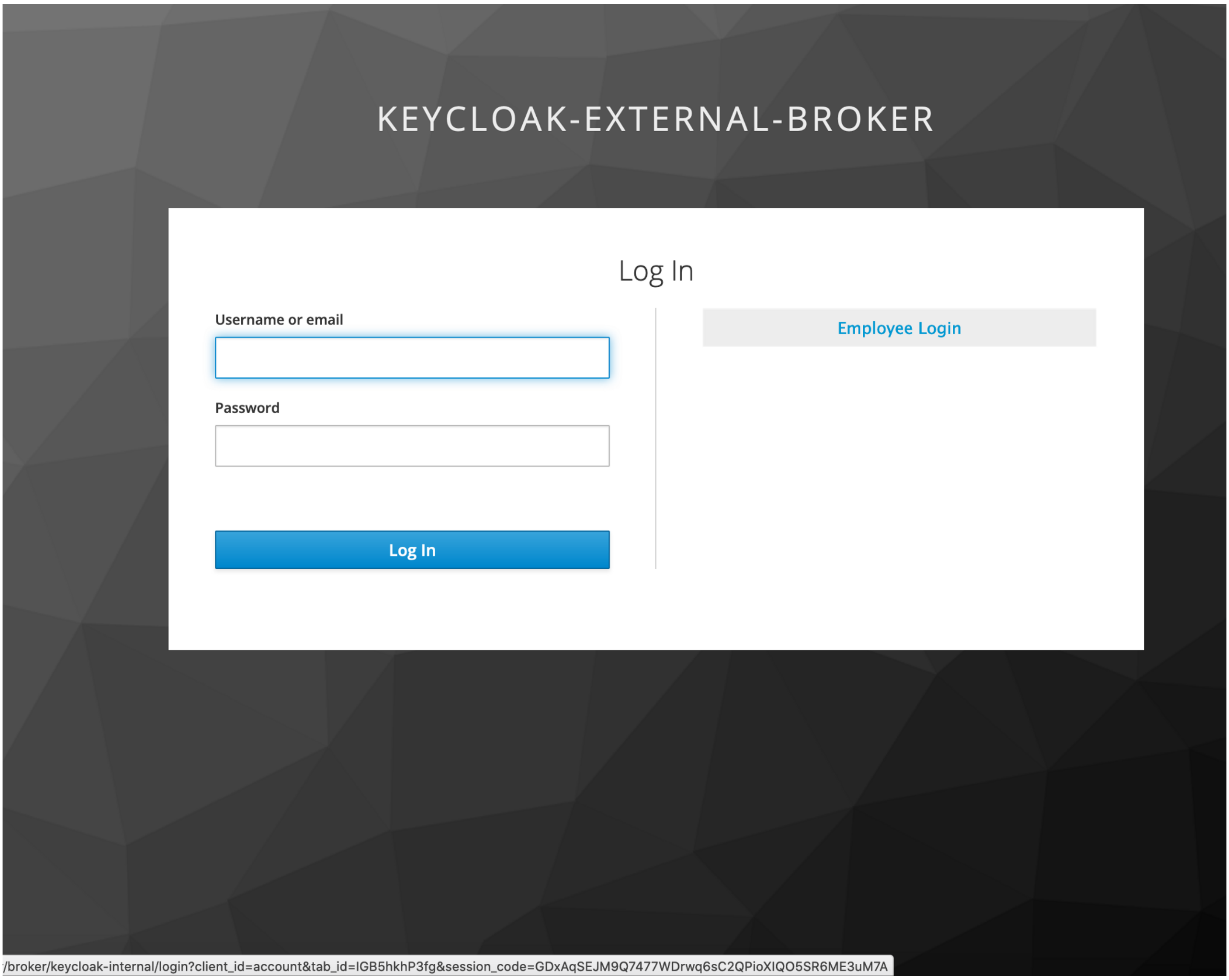
- Update the Valid Redirect URI:

```
http://<host>:<ip>/auth/realms/<realm>/broker/keycloak-
internal/endpoint
```

That’s all you need. Create a demo user from the user's section in Keycloak Internal (127.0.0.1:8080).

Now visit URL: External Account login

```
http://127.0.0.1:8081/auth/realms/keycloak-external-broker/account
```



KEYCLOAK-EXTERNAL-BROKER

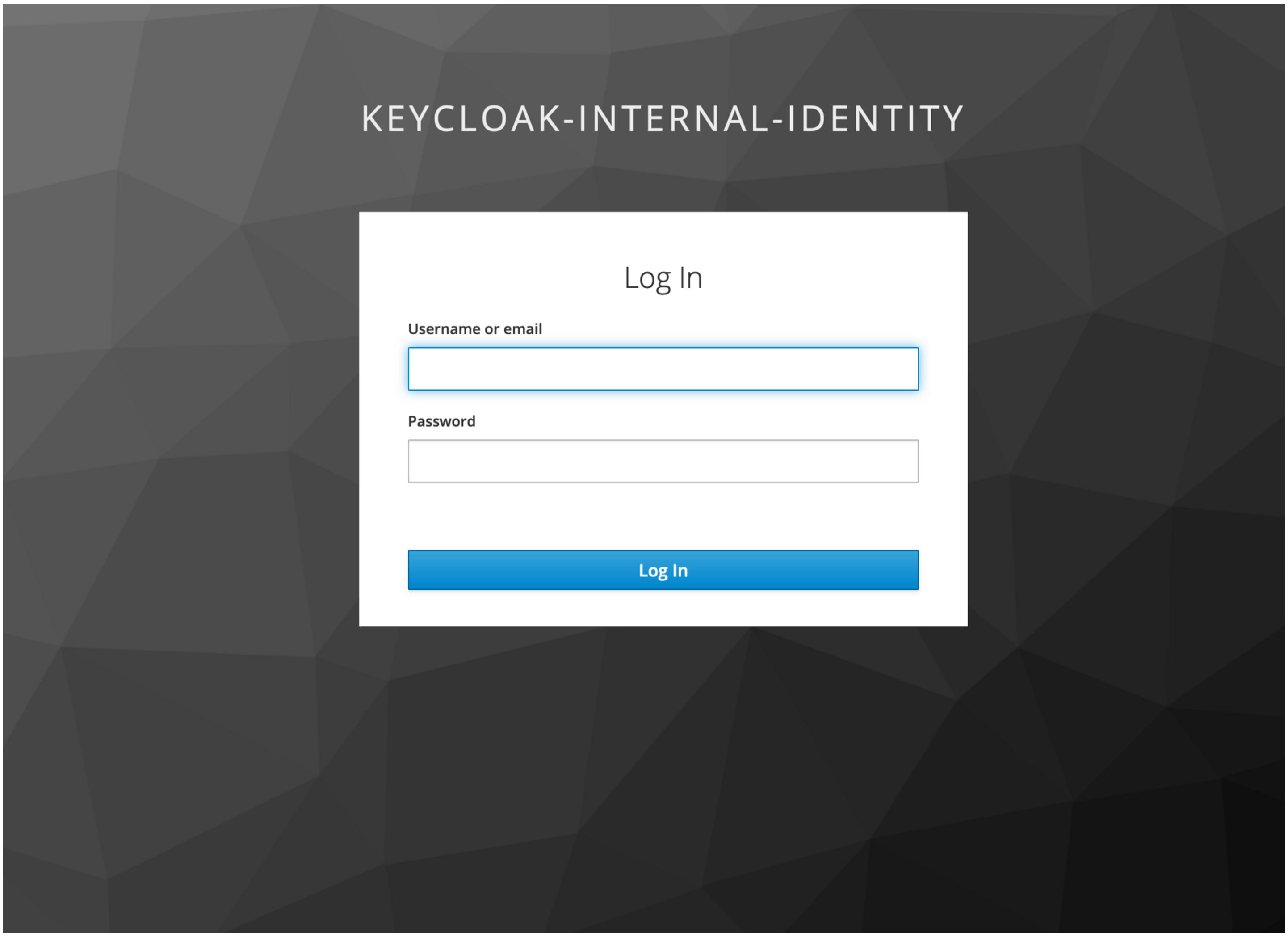
Log In

Username or email

Password

/broker/keycloak-internal/login?client_id=account&tab_id=IGB5hkhP3fg&session_code=GDxAqSEJM9Q7477WDrwq6sC2QPioXIQO5SR6ME3uM7A

We will use the “Employee Login” option for internal users. It will redirect to the internal Keycloak instance.



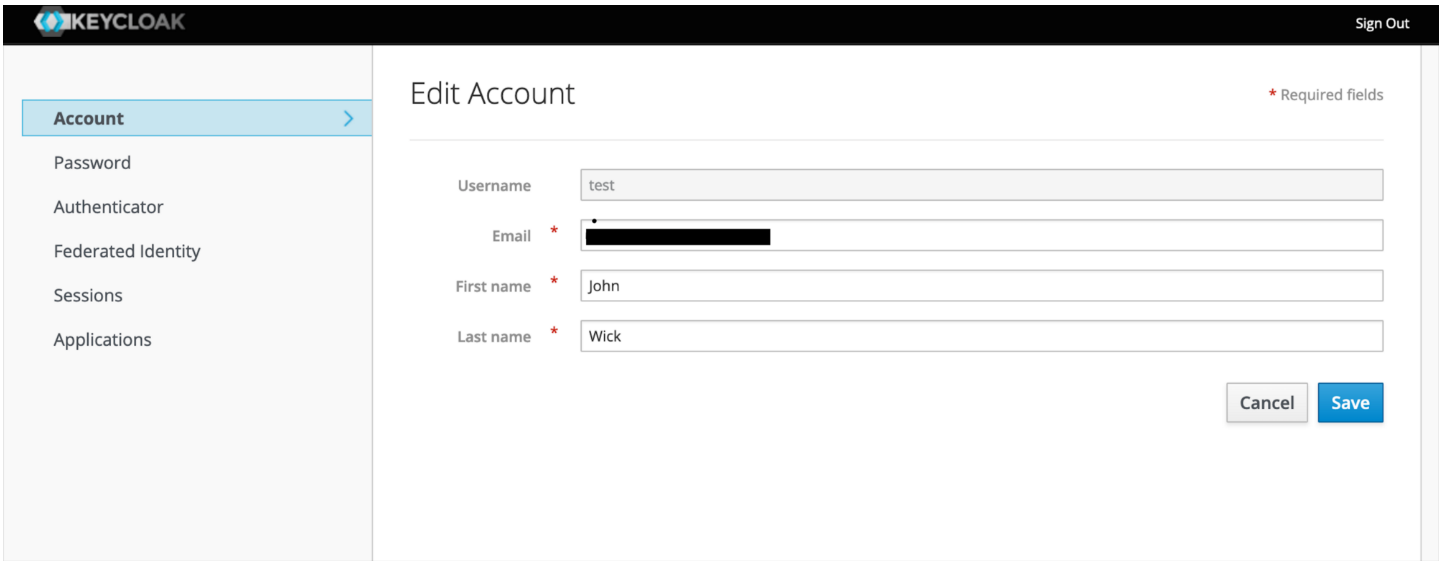
KEYCLOAK-INTERNAL-IDENTITY

Log In

Username or email

Password

Now you can see the User: test is logged into external Keycloak.



Edit Account

* Required fields

Username: test

Email *: [redacted]

First name *: John

Last name *: Wick

Conclusion

Brokering & Identity provider is a powerful pattern that can help deal with the most complex problem of handling authentication & authorization. Keycloak supports both & allow to extend further with a custom implementation.

If you like this post, give a Cheer!!!

Follow the Collection: Keycloak for learning more…

Happy Secure Coding ❤

Broker Identity Keycloak Openid Connect