Keycloak with Supabase setup

Guide: <https://supabase.com/docs/guides/auth/social-login/auth-keycloak>

NOTE: might not work in Docker/locally

Steps:

* Login as admin in e.g. <http://localhost:8080> where Keycloak is setup
* Create realm, define name for the realm
  + Realm manages a set of users, credentials, roles and groups
  + Isolated from one another and can only manage users they control
* Create user in Users tab
  + Add username
  + Go to credentials -> set password and set Temporary: Off
  + Login as user in [http://localhost:8080/realms/<realm-name>/account/](http://localhost:8080/realms/%3crealm-name%3e/account/)
  + On first login you have to setup email, first name and last name if not already set
* Create client from Clients tab
  + Type: OpenID Connect
  + ID: id for the client
  + Client authentication ON for OIDC confidential access type, OFF for OIDC public access type (OFF for client-side applications / frontend)
  + Authentication flow should include Standard flow (OpenID Connect redirect based authentication with authorization code)
* Valid redirect URIs should include Supabase callback URL
  + In Supabase, go to Authentication -> Providers -> KeyCloak
  + Copy-paste Callback URL into KeyCloak’s Valid redirect URIs
    - Usually https://<your-supabase-url>.supabase.co/auth/v1/callback
* Retrieve issuer from OpenID EndPoint configuration
  + [http://localhost:8080/realms/<my\_realm\_name>/.well-known/openid-configuration](http://localhost:8080/realms/%3cmy_realm_name%3e/.well-known/openid-configuration)
  + Most likely in form <ip-address>/realms/<realm-name>
* Obtain client secret
  + Credentials -> client secret
* In Supabase add client ID, client secret and realm URL (from issuer)
* Test Keycloak in Postman
  + Make POST request to <ip-addr>/realms/<realm-name>/protocol/openid-connect/token with body in x-www-form-urlencoded with content
    - client\_id: <my-client-id>
    - username: <my-username>
    - password: <my-password>
    - grant\_type: password
  + A screenshot of a computer

    Description automatically generated
  + On success you should receive access token and refresh token
* Test Keycloak in React
  + With supabase-js: <https://supabase.com/docs/guides/auth/social-login/auth-keycloak>
  + OR
  + Keycloak JS adapter for client-side:
  + <https://www.keycloak.org/docs/latest/securing_apps/index.html#_javascript_adapter>
    - npm install keycloak-js
    - Can also npm install @react-keycloak/web
    - Can use example here: <https://blog.logrocket.com/implement-keycloak-authentication-react/>
      * Might need to omit “auth” from URLs
    - <https://www.geeksforgeeks.org/how-to-implement-keycloak-authentication-in-react/>
  + Keycloak Node.js adapter for server-side apps:
  + <https://www.keycloak.org/docs/latest/securing_apps/index.html#_nodejs_adapter>

Our Keycloak Supabase setup

In Pouta server: <https://keycloak.ilab.fi:8443/>

* Realm: fullstackauth
* Client: supabase
  + Type: OpenID Connect
  + ID: supabase
  + Client authentication: Off
  + Authentication flow: Standard flow, Direct access grants (change if needed)
* Valid redirect URIs & Web origins
  + <https://bmfjjujhrtssdmcbnrio.supabase.co/auth/v1/callback>
  + [http://localhost:5173/\*](http://localhost:5173/*)
  + (<https://www.authzilla.ilab.fi/> not implemented yet)
* Issuer from [keycloak.ilab.fi:8443/realms/fullstackauth/.well-known/openid-configuration](https://keycloak.ilab.fi:8443/realms/fullstackauth/.well-known/openid-configuration)
  + <https://keycloak.ilab.fi:8443/realms/fullstackauth>
* Test user is test:test
* User login from <https://keycloak.ilab.fi:8443/realms/fullstackauth/account/>
* Using supabase-js library in React
* NOTE!
  + For now, client authentication is set OFF
    - If ON: Token response: 401 unauthorized, error: “unauthorized client”, error\_description: “Invalid client or invalid client credentials”
    - Using backend could fix so it would be properly secure because frontend is a public client