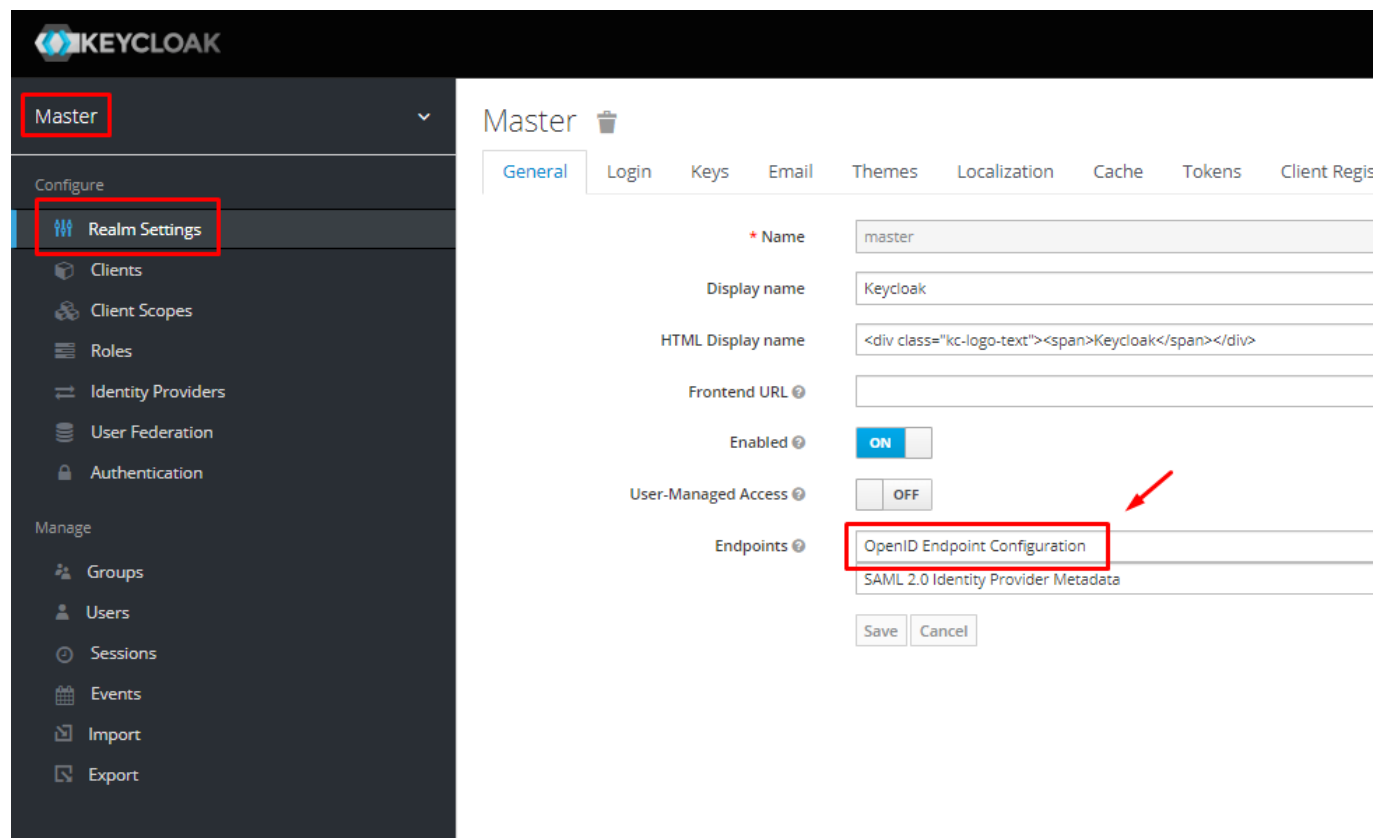


_1. Conocer los endpoint de **OpenID OIDC** en **Keycloak**, para esto seleccionamos el realm «**master**», Configure «**Realm Settings**» y en la pestaña «**General**» damos clic donde dice «**OpenID Endpoint Configuration**»



The screenshot shows the Keycloak Admin Console interface. On the left sidebar, the 'Master' realm is selected, and 'Realm Settings' is highlighted under the 'Configure' section. The main content area shows the configuration for the 'master' realm in the 'General' tab. The 'Endpoints' section is visible, with 'OpenID Endpoint Configuration' highlighted by a red box and a red arrow pointing to it. Other visible fields include 'Name' (master), 'Display name' (Keycloak), 'HTML Display name' (HTML code), 'Frontend URL', 'Enabled' (ON), and 'User-Managed Access' (OFF).

Se abrirá una URL <http://localhost:8080/realms/master/.well-known/openid-configuration>

```
1 // 20220708233925
2 // http://localhost:8080/realms/master/.well-known/openid-configuration
3
4 {
5   "issuer": "http://localhost:8080/realms/master",
6   "authorization_endpoint": "http://localhost:8080/realms/master/protocol/openid-connect/auth",
7   "token_endpoint": "http://localhost:8080/realms/master/protocol/openid-connect/token",
8   "introspection_endpoint": "http://localhost:8080/realms/master/protocol/openid-connect/token/introspect",
9   "userinfo_endpoint": "http://localhost:8080/realms/master/protocol/openid-connect/userinfo",
10  "end_session_endpoint": "http://localhost:8080/realms/master/protocol/openid-connect/logout",
11  "frontchannel_logout_session_supported": true,
12  "frontchannel_logout_supported": true,
13  "jwks_uri": "http://localhost:8080/realms/master/protocol/openid-connect/certs",
14  "check_session_iframe": "http://localhost:8080/realms/master/protocol/openid-connect/login-status-iframe.html",
15  "grant_types_supported": [
16    "authorization_code",
17    "implicit",
18    "refresh_token",
19    "password",
20    "client_credentials",
21    "urn:ietf:params:oauth:grant-type:device_code",
22    "urn:openid:params:grant-type:ciba"
23  ],
24  "acr_values_supported": [
25    "0",
26    "1"
27  ],
28  "response_types_supported": [
29    "code",
30    "none",
31    "id_token",
32    ". . . "
```

2. Nos vamos a **Postman** y consultamos el token, donde:

- **Token method:** POST
- **Token endpoint:** http://localhost:8080/realms/master/protocol/openid-connect/token
- **x-www-form-urlencoded**
 - **grant_type:** password
 - **client_id:** admin-cli (es la aplicación cliente registrada en keycloak, se puede crear otra aplicación)
 - **username:** juliopari (previamente creado)
 - **password:** 123456 (previamente creado)

The image shows a web interface for decoding a JWT token. On the left, under 'Encoded', there is a text area containing a long base64-encoded string. On the right, under 'Decoded', the token is shown in its structured form. The header indicates the algorithm is 'RS256' and the type is 'JWT'. The payload contains user data: 'exp', 'iat', 'jti', 'iss', 'sub', 'typ', 'azp', 'session_state', 'acr', 'scope', 'sid', 'email_verified', 'name', 'preferred_username', 'given_name', 'family_name', and 'email'.

Julio Pari (IT Architect IBM)



Si te ha interesado este artículo y deseas un apoyo o asesoría en algún requerimiento, envíame un mensaje a: (info@juliopari.com) o sino a través de LinkedIn: <https://www.linkedin.com/in/juliopari/>