Realms

Деление на блоки через realm. Каждый realm изолирован, пользователь принадлежит только одному realm. Содержит конфигурацию, набор приложений и пользователей. Есть административный и остальные realm.

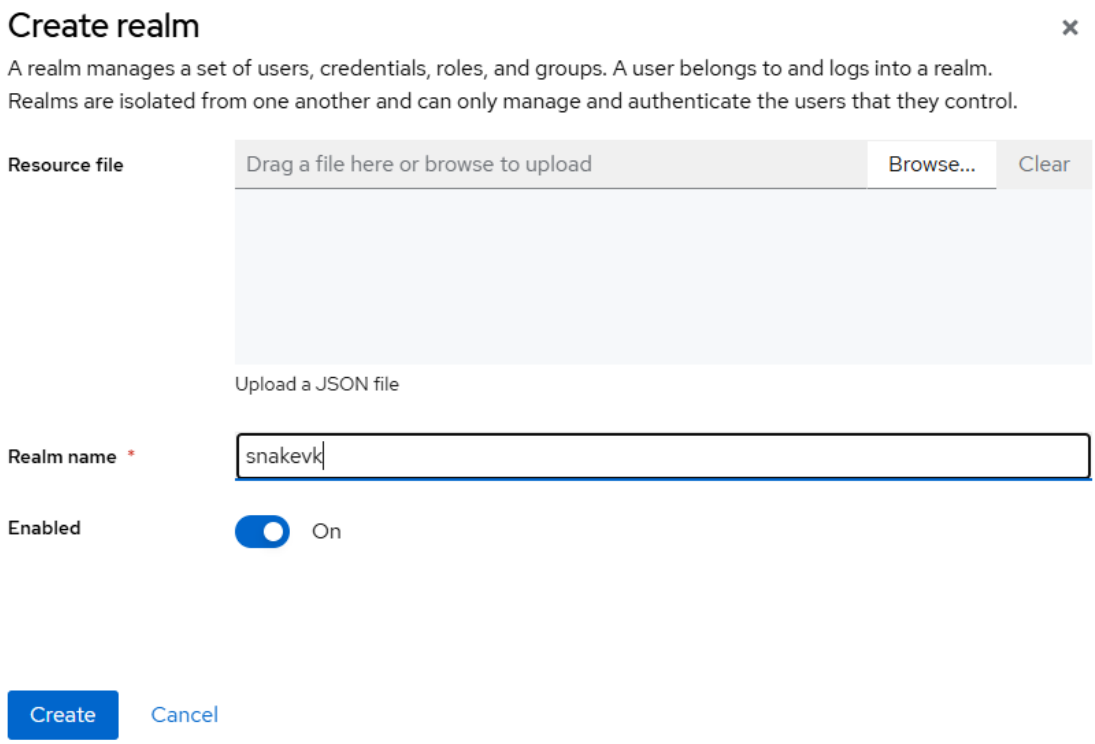
Master: административный realm, задачи:

- Управления другими realms (создание, удаление и настройка realms, управление пользователями-администраторами)
- Создание глобальных админов. Роли realm-admin, admin, create-realm и т. д. позволяют входить в Keycloak Admin Console (/admin) и управлять сервером.
- Использование Keycloak'ом для внутренних задач. Системные клиенты и сервисные аккаунты находятся в master realm.
  Примеры: admin-cli, account, broker.

Есть консоль управления реалм.

Создание realm

Настройка realm в момент создания - только имя

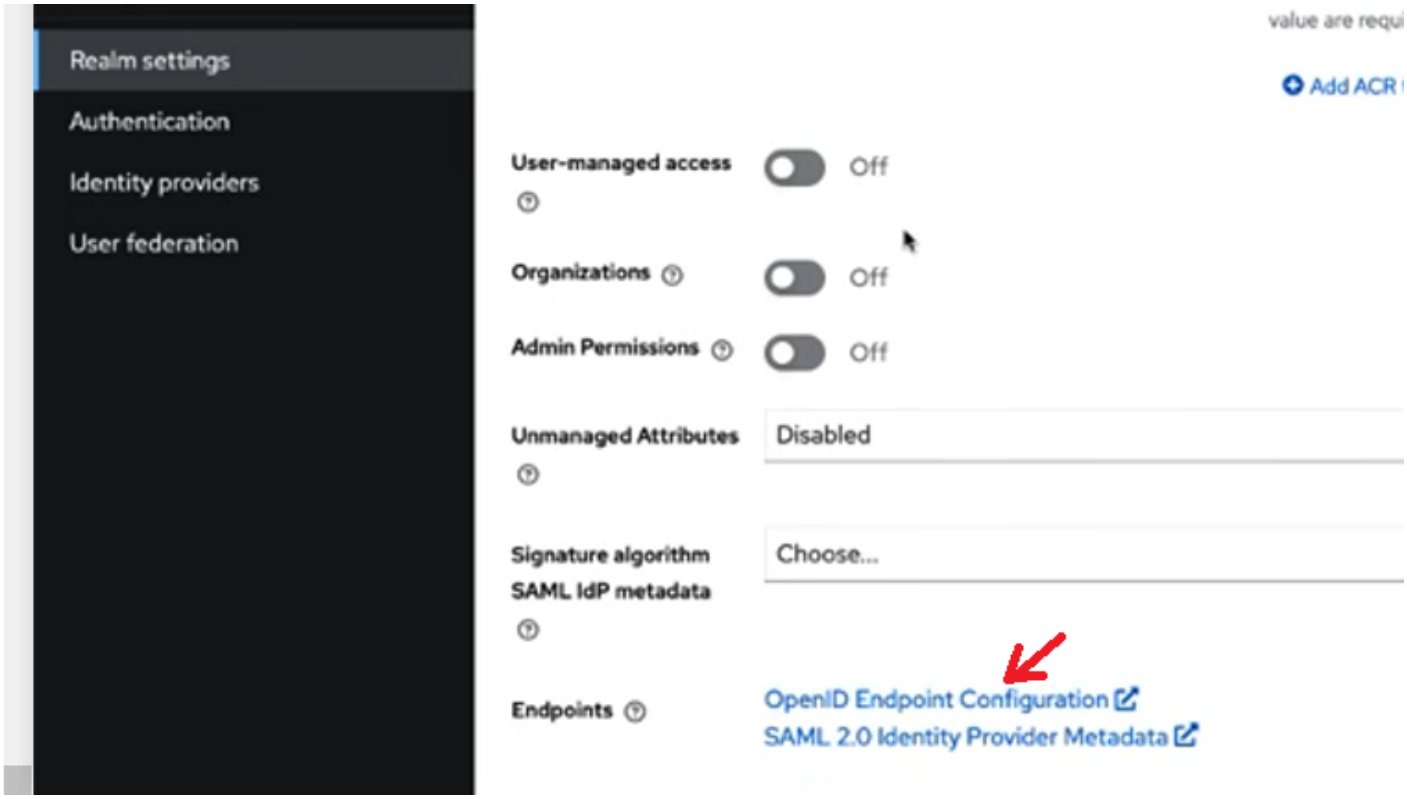


Имя не должно содержать пробелов и иных символов, плохо воспринимаемых в url адресе.

Настройка realm (realm settings)

В меню Manage realms выбирается реалм, который мы будем редактировать. Затем в левом меню, раздел Configure содержит 4 блока.

OpenID Endpoint configuration - здесь можно увидеть все endpoint настроенные для данного realm.



Настройка процесса аутентификации. Раздел Authentication используется для настройки и управления потоками аутентификации, обязательными действиями, и политиками входа. Authentication flow (поток аутентификации) - это набор шагов, через которые проходит пользователь или клиент при входе, регистрации, сбросе пароля и т.д. Каждый поток состоит из действий (executions): например, проверка пароля, OTP, условные переходы, выбор IdP и др.

Потоки создаются и настраиваются.

Grant Type - способ, получения Access Token от сервера авторизации в OAuth 2.0 / OpenID Connect. Тип разрешения (grant) определяет, как и при каких условиях клиент получает токены. Keycloak, как реализация Identity Provider и Authorization Server, поддерживает разные grant types (,).

Название	Описание	Тип приложения

Authorization Code	Стандартный безопасный flow для веб-приложений, где сервер клиента может хранить секреты. Клиент формирует POST-запрос к token-эндпоинту авторизационного сервера с обязательными параметрами: • grant_type -> Authorization Code • authorization_code -> code, полученный на шаге 6 • redirect_uri -> тот же URI, что использовался при авторизации • client_id -> Идентификатор клиента • client_secret -> Секрет клиента безопасно для серверов с хранением Client Secret. Уязвимость на этапе редиректа. Подходит для серверных приложений	Веб-приложения (backend + frontend).

Authorization Code + PKCE	Модификация Authorization Code Flow для публичных клиентов (например, мобильных и SPA), чтобы предотвратить атаку перехвата кода. Клиент перед началом авторизации: • Генерирует случайный код (code_verifier) • Строит на его основе code_challenge (обычно SHA-256) • При первом запросе отправляется code_challenge • При обмене Authorization Code на Access Token отправляется code_verifier. • Authorization Server сверяет code_challenge и code_verifier: Если всё сходится, только тогда выдает Access Token. Это защищает от атаки перехвата кода, потому что даже если злоумышленник украдет Authorization Code, без правильного code_verifier он не сможет обменять его на Access Token. безопасно для мобильных и браузерных клиентов без секретов. Нужен только Code Verifier Защита на этапе редиректа Подходит для мобильных и SPA	Мобильные приложения, SPA (Single Page Applications).
Client Credentials	Классический machine-to-machine сценарий, когда приложение запрашивает токен от собственного имени, без пользователя.	Сервисные взаимодействия (backend to backend), микросервисы.
Device Code	Для устройств без полноценной клавиатуры (например, Smart TV). Пользователь вводит код на другом устройстве.	Устройства с ограниченным вводом - SmartTV, консоли, IoT.
Refresh Token	После получения Access Token можно получить новый токен без повторной аутентификации пользователя.	Любые клиенты с долгоживущими сессиями.

Authentication

Настройка потоков аутентификации, обязательных действий и политик входа.
Authentication flow (поток аутентификации) - это набор шагов, через которые проходит пользователь или клиент при входе, регистрации, сбросе пароля и т.д.
Каждый поток состоит из действий (executions): например, проверка пароля, OTP, условные переходы, выбор IdP и др.

Название	**Описание**
browser	Встроенный поток для авторизации на основе браузера. Включает формы логина, OTP (двухфакторную аутентификацию), CAPTCHA и другие шаги.
clients	Встроенный поток аутентификации для клиентов (используется в client credentials grant). Работает без участия пользователя.
direct grant	Встроенный поток для ресурсного владельца (Resource Owner Password Credentials Grant). Используется, когда пользователь напрямую вводит логин и пароль (например, через curl или мобильное приложение).
docker auth	Поток аутентификации Docker-клиентов при доступе к приватному Docker Registry через Keycloak.
First broker login	Поток, срабатывающий при первом входе пользователя через внешний Identity Provider (например, Google, GitHub). Позволяет привязать внешний аккаунт к локальному пользователю Keycloak.
Registration	Поток регистрации новых пользователей. Включает форму ввода данных, подтверждение по email и проверки по политике безопасности (например, сила пароля).
Reset credentials	Поток для восстановления доступа, если пользователь забыл пароль. Может включать подтверждение по email, OTP и другие шаги для подтверждения личности.

У каждого realm свой открытый ключ. С его помощью можно проверять подлинность полученных токенов. После авторизации под любым пользователем выполнить один раз

```
KEYCLOAK_PUBLIC_KEY = f"""
-----BEGIN PUBLIC KEY-----
{keycloak_openid.public_key()}
-----END PUBLIC KEY-----
"""
```

И вывести значение. Это будет открытым ключом.

User Federation

Подключение внешних источников пользователей (LDAP, Active Directory или Kerberos).

- пользователи остаются в LDAP/AD/Kerberos;
- Keycloak может их аутентифицировать;
- можно синхронизировать или кэшировать данные (имя, email, роли и т.д.);
- поддерживается одноразовый вход (SSO), если внешняя система это позволяет.

SAML

Security Assertion Markup Language - открытый стандарт SSO между разными доменами. Позволяет аутентифицироваться в одной системе и получить доступ к другой, предоставив подтверждение своей аутентификации. Используется с 2005 года и остаётся популярным для федерации идентичности в B2B и B2E-сценариях, особенно в государственном и корпоративном секторе. SAML оперирует:

- XML - для описания данных пользователя
- HTTP - как транспортный протокол

Revision #10
Created 11 April 2026 06:38:38 by Admin
Updated 20 May 2026 15:26:25 by Admin