

- 1. Інтеграція keycloak з vue.js, Node.js express в openshift
- 2. Розгортання keycloak в OpenShift
- 3. Основні терміни адміністрування Keycloak
- 4. Ручна реєстрація програми клієнта
- 5. Налаштування прикладних ролей для програми клієнта
- 6. Створення користувачів для призначення їм ролей
- 7. Отримання авторизаційного токена по протоколу openid-connect

Інтеграція keycloak з vue.js, Node.js express в openshift

Продукт [Keycloak](#) є зараз типовим інструментом для авторизації в Web-based системах. Документацію можна знайти за лінком:

- Основна документація знаходиться за лінком [documentation](#);
- За цим лінком знаходяться більш фокусовані описи [Фокусовані описи](#).

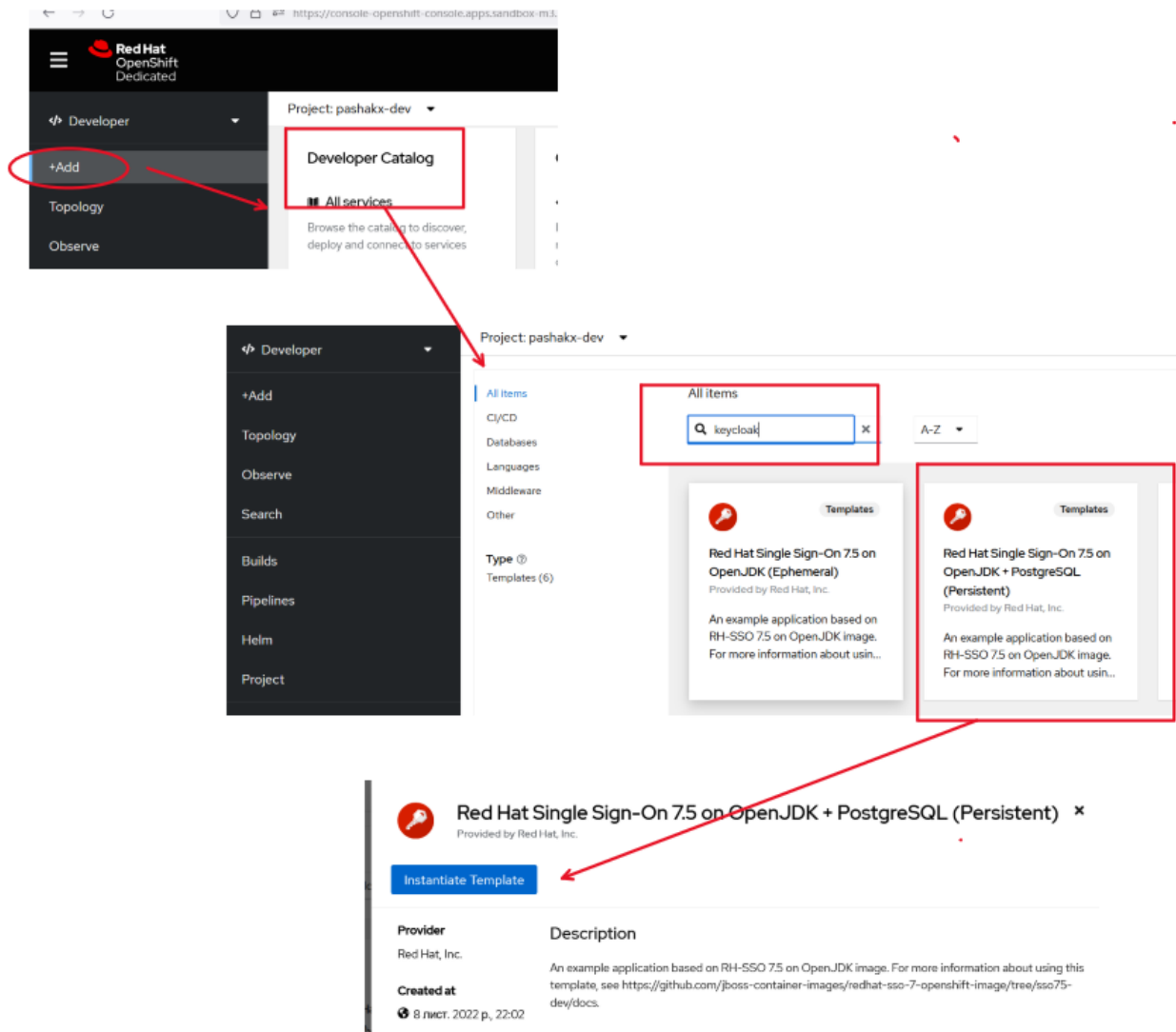
Ну на цьому зацікавився цим продуктом, щоб трохи розібратися як він працює та як його конфігурувати.

Розгортання keycloak в OpenShift

За звичай keycloak можна підняти в контейнері, але він стартує в development mode. Так також там треба прив'язати базу даних типу postgresql. Я вирішив піти більш простим шляхом і розгорнув його в хмарі RedHat в sandbox OpenShift. Openshift sandbox можна створити за url:

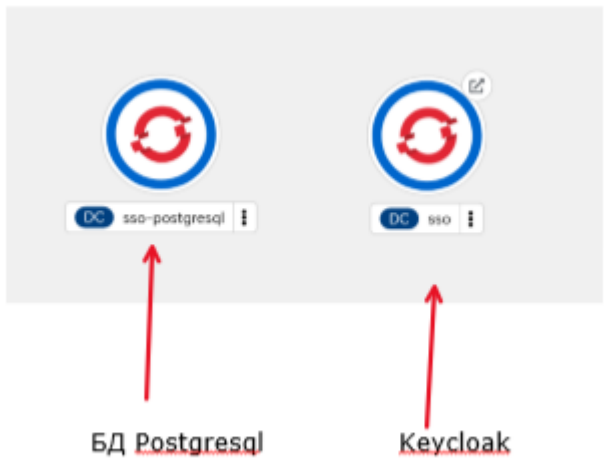
<https://developers.redhat.com/developer-sandbox>, а перед цим потрібно зареєструватися як developer на RedHat. Docker відкинув зразу, тому що прочитав ліцензійні обмеження для корпорацій.

Щоб не мучитися з лінуванням бази даних та самого сервісу keycloak я використав калог уже підготованих продуктів, що є вже в будь-якому OpenShift і в пару кліків розгорнув додаток [pic-01](#).



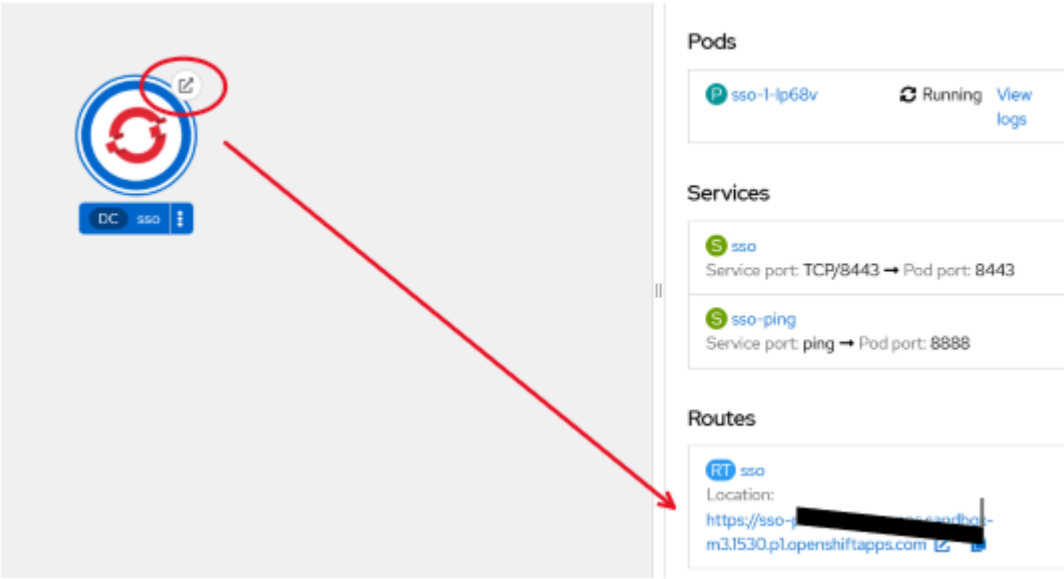
pic-01

Через кілька хвилин я уже маю готовий keycloak:



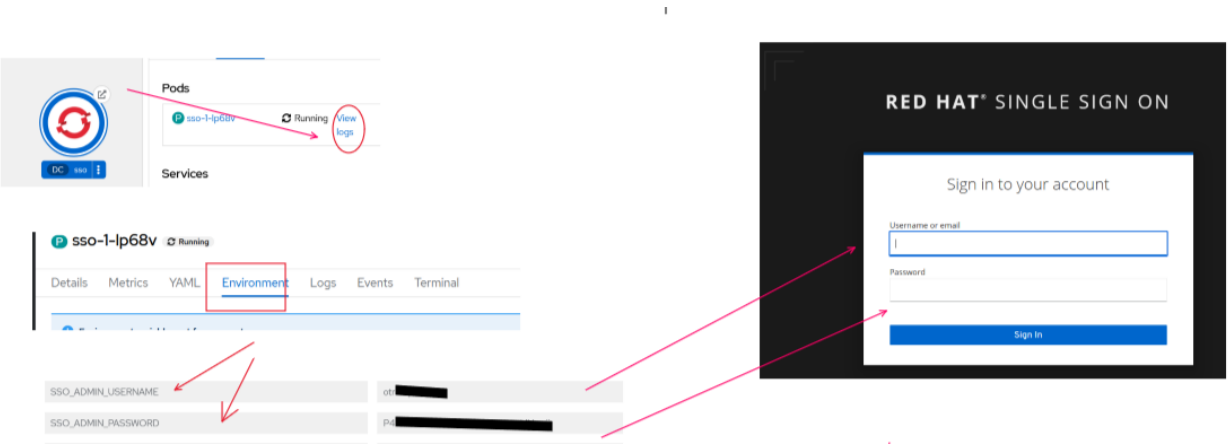
pic-02

І уже натискаємо на роут [pic-03](#), попадаємо в консоль адміністрування. Можливо потрібно трохи зачекати, поки обидва сервіса стартонуть. Ну, там час старту такий собі, відчутний, але не критично.



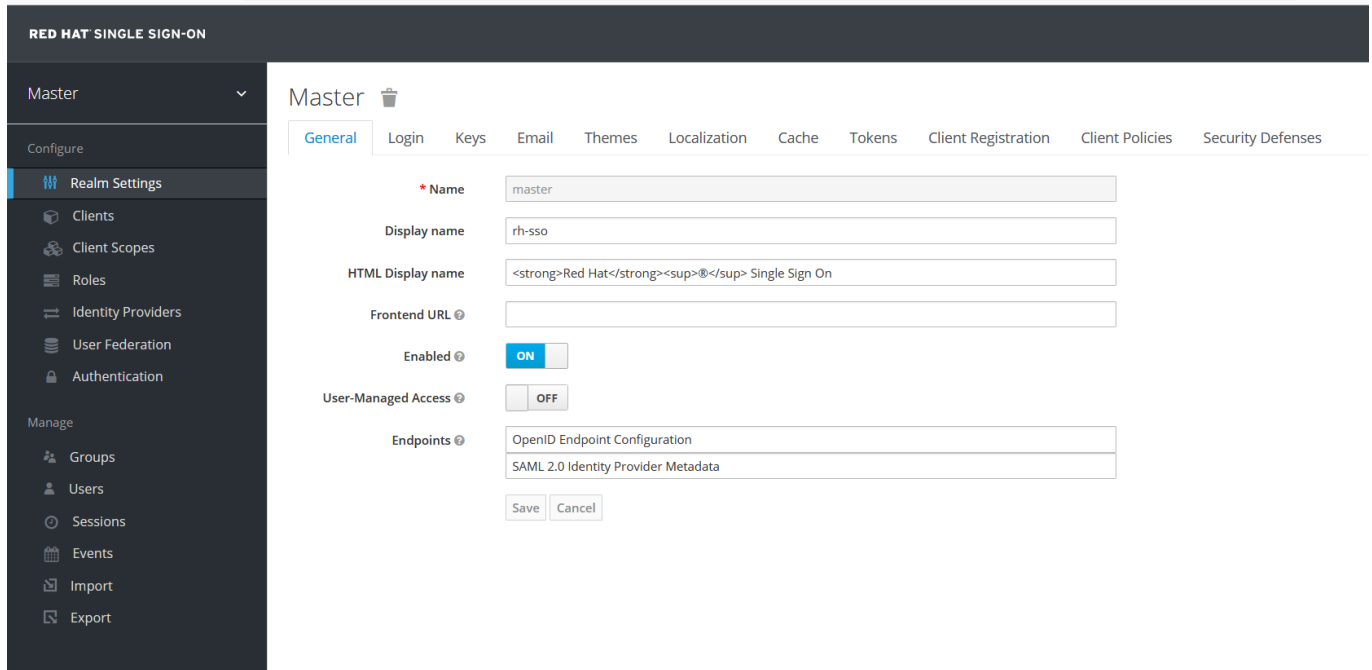
pic-03

Консоль адміністрування запросить логін та пароль. Вони знаходяться в env змінних Keycloak так як показано на [pic-04](#)



pic-04

Залогінився і вуаля - попадаємо в консоль адміністрування, в головний realm

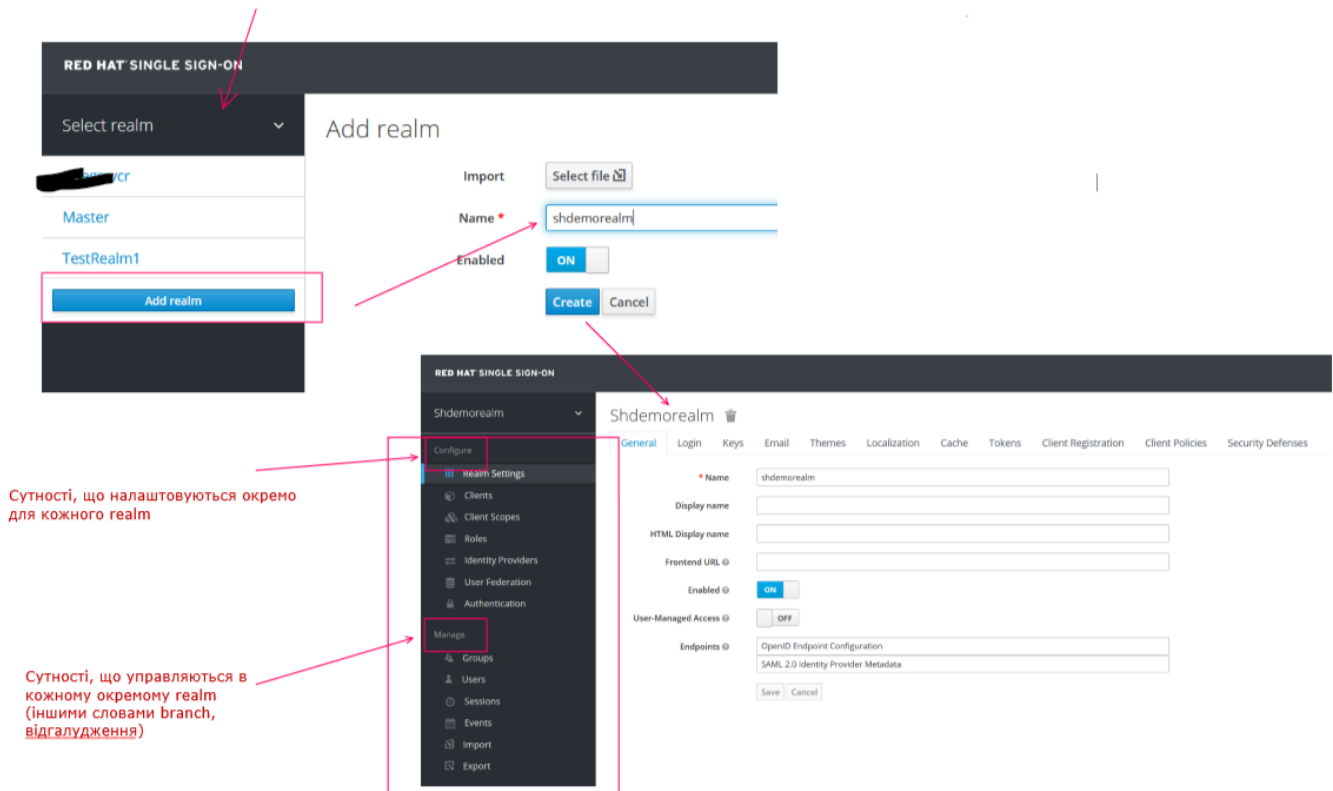


pic-05

Основні терміни адміністрування Keycloak

Адміністрування в keycloak ділиться на **realm**-и.

- **Realm** це одиниця адміністрування, що інкапсулює в собі набори: користувачів, ролей, груп та набори додаткових повноважень (чи прав). Тому, перше, що робимо, реєструємо свій realm. Назвемо його **shdemorealm**. На [pic-06](#) показані основні елементи realm.



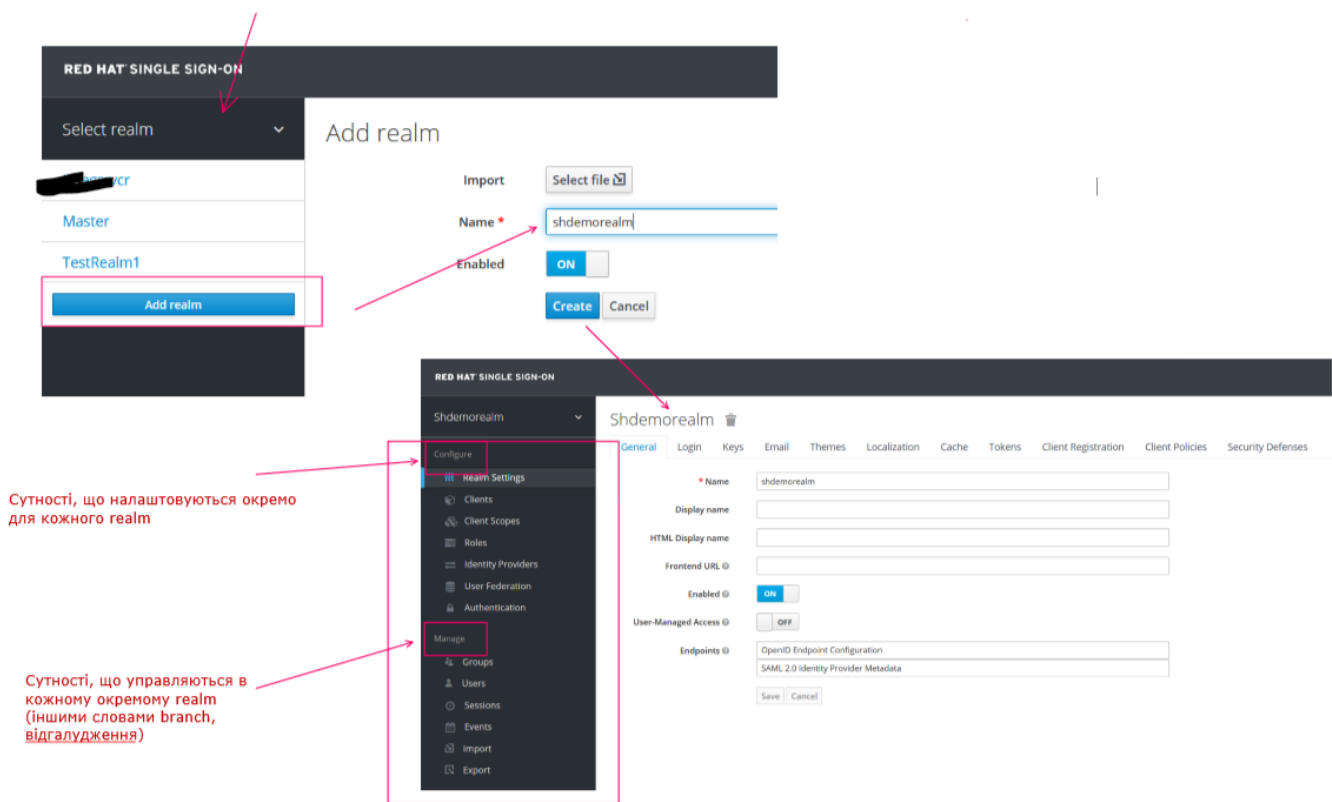
pic-06

Keycloak підтримує як OpenID Connect (розширення OAuth 2.0), так і SAML 2.0. Коли говоримо про security, перше, що потрібно вирішити, це те, що з двох ви збираєтеся використовувати. Я вирішую використовувати OpenID Connect (розширення OAuth 2.0), тому в подальшому мова йде тільки про нього.

- **Client** - це прикладні додатки, що зареєстровані в окремо взятій realm. За звичай додатки аутентифікуються за допомогою client ID або client ID та client secret. Ще є аутентифікація через JWT токен, але зараз її не розглядаємо. Client ID and Client Secret - це традиційний метод, описаний у специфікації OAuth2. Клієнт має секрет, який повинен бути відомий як прикладному додатку, так і серверу Keycloak.

Ручна реєстрація програми клієнта

Створюємо клієнта в client ID **DemoApp1**, як показано на [pic-06](#):



pic-06

Після реєстрації зразу попадаємо на вікно [pic-07](#)

DemoApp1

Settings Keys Roles Client Scopes Mappers Scope Revocation Sessions Offline Access Installation

Client ID DemoApp1

Name

Description

Enabled ☒

Always Display in Console ☐

Consent Required ☐

Login Theme

Client Protocol openid-connect

Access Type public

Standard Flow Enabled ☒

Implicit Flow Enabled ☐

Direct Access Grants Enabled ☒

Client Protocol ?

Access Type ?

Standard Flow Enabled ?

Implicit Flow Enabled ?

Direct Access Grants Enabled ?

'Confidential' clients require a secret to initiate login protocol. 'Public' clients do not require a secret. 'Bearer-only' clients are web services that never initiate a login.

This enables standard OpenID Connect redirect based authentication with authorization code. In terms of OpenID Connect or OAuth2 specifications, this enables support of 'Authorization Code Flow' for this client.

pic-07

Тут треба звернути увагу на **Access Type**=public при логіні не поребує client Secret. А при **Access Type**=confidential потребує client Secret. Зараз залишаємо **public**.

Треба звернути увагу на **Standard Flow Enabled** треба включити в "ON". Таким чином авторизація піде по стандартному OAuth2.0 за допомогою обміну authorization_code.

Далі потрібно налаштувати URL додатка так, як показано на [pic-08](#).

Root URL http://localhost:8080

* Valid Redirect URIs http://localhost:8080/*

Base URL

Admin URL

Web Origins

Backchannel Logout URL

pic-08

Тобто, поки що нас цікавлять Root URL та "правильний" redirect URL. По ньому відбувається переадресація у випадку успішного логіна. Інші, пока що не задіяні.

Все, можна сказати в найпростішому варіанті клієнт-додаток зареєстровано.

Налаштування прикладних ролей для програми клієнта

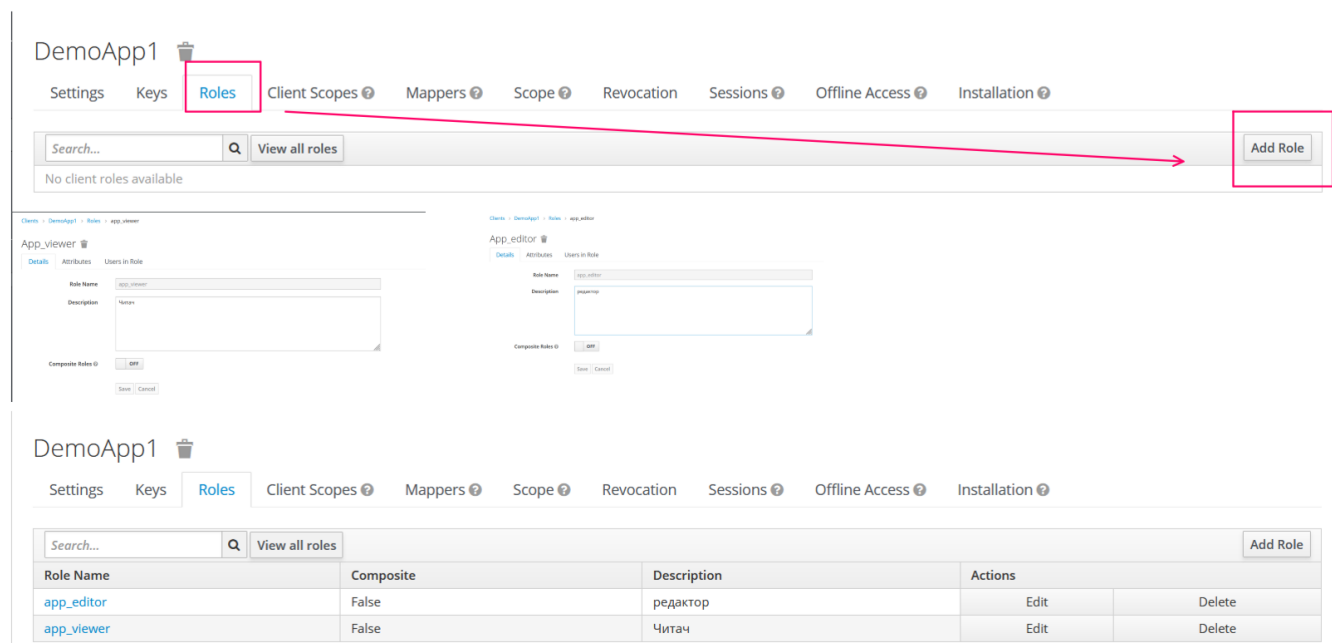
Після реєстрації додатку-клієнта потрібно запараметризувати рольовий доступ до ресурсів програми. Рольовий досутп може ділитися на по ролям. А ролі діляться на Realm Roles та Client Roles.

- Realm Roles доступні для всього realm. Цей варіант не зовсім вигідний. Але прийнятний в деяких випадках
- Client Roles - доступні для окремо зареєстрованого клієнта. Цей варіант, як на мене, більш прийнятний. Тому на ньому і зупинимося.

Для нашого майбутнього додатку створемо 2 ролі:

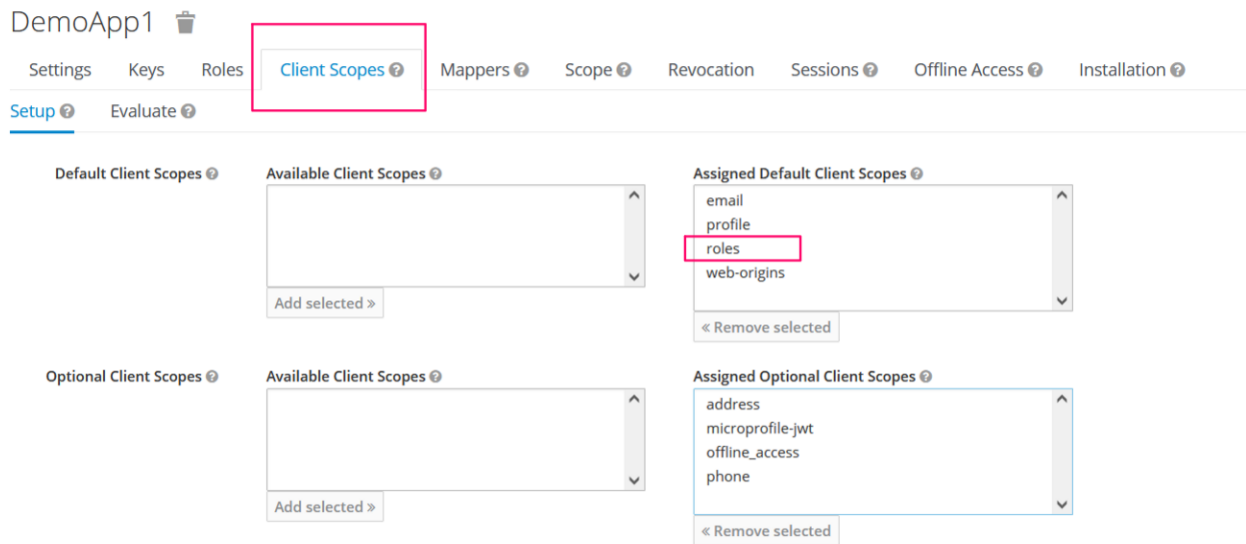
- app_viewer дозволяє тільки читати дані:
- app_editor дозволяє читати та змінювати дані.

Для цього створюємо ролі, як показано на [pic-09](#)



pic-09

Щоб ролі передавалися на клієнта, потрібно пересідчитися, що вони включені в client scope, як на [pic-10](#).

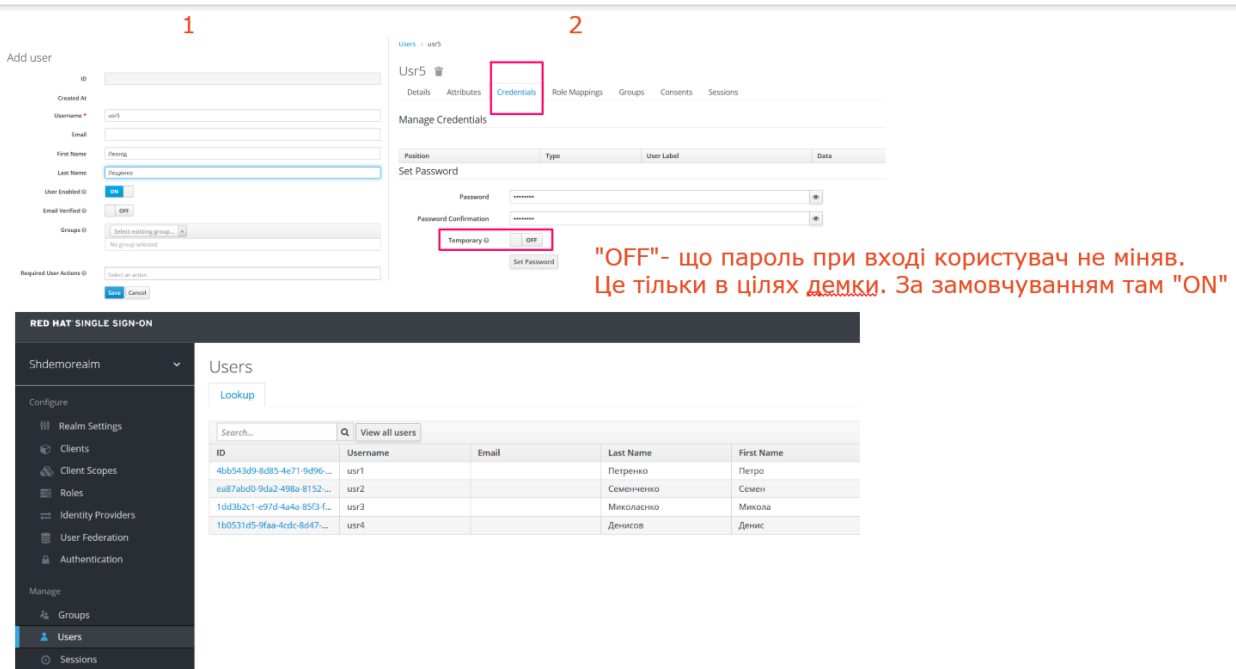


pic-10

Створення користувачів для призначення їм ролей

Набір користувачів є єдиним на весь realm. Але користувачів можна розділити на групи. Набір груп теж єдиний на весь realm. А от групи можна вже поєднати з ролями. Так і зробимо.

Процес створення користувачів показано на [pic-11](#).



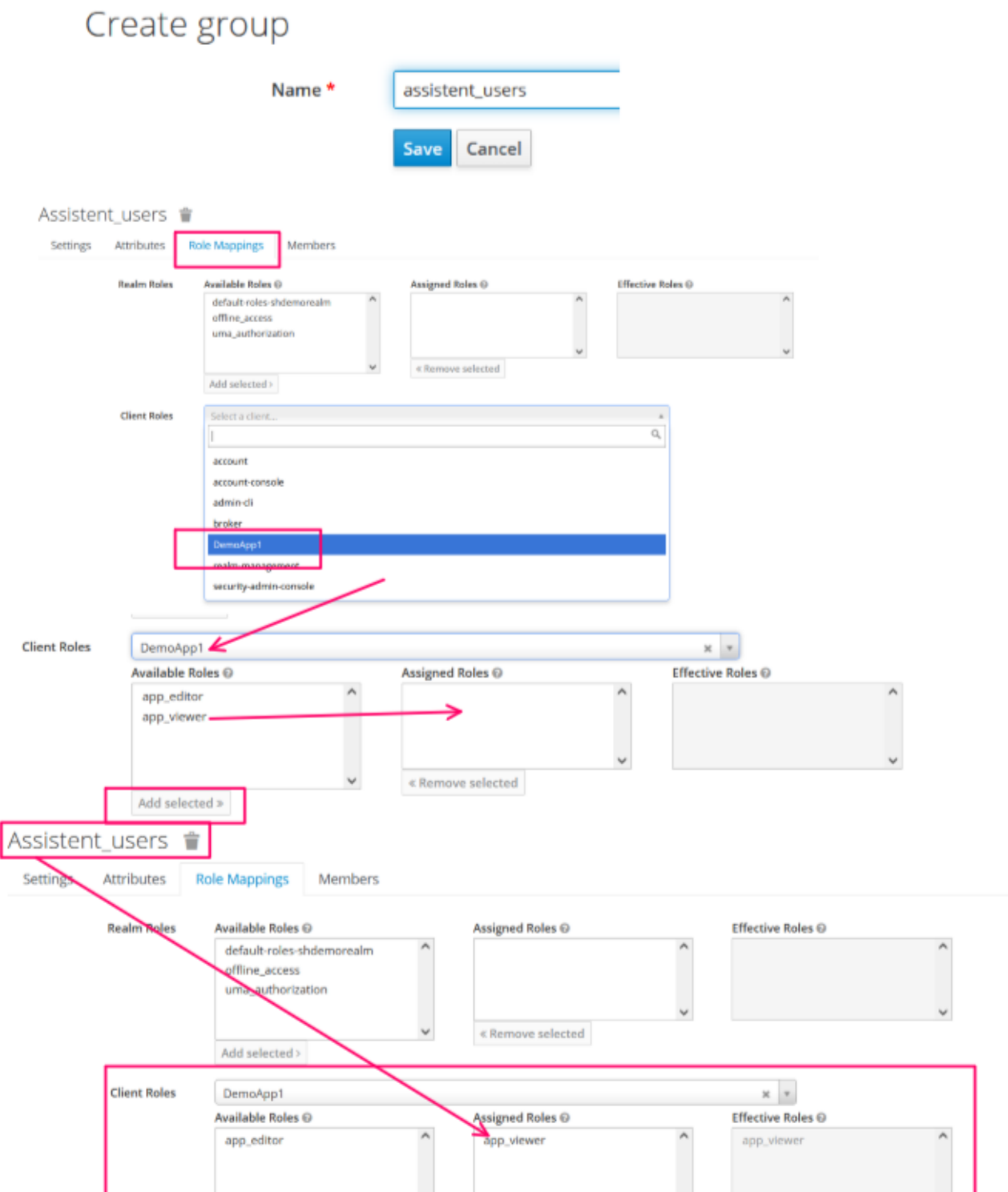
pic-11

Тепер створимо групи користувачів і поділимо користувачів на дві групи:

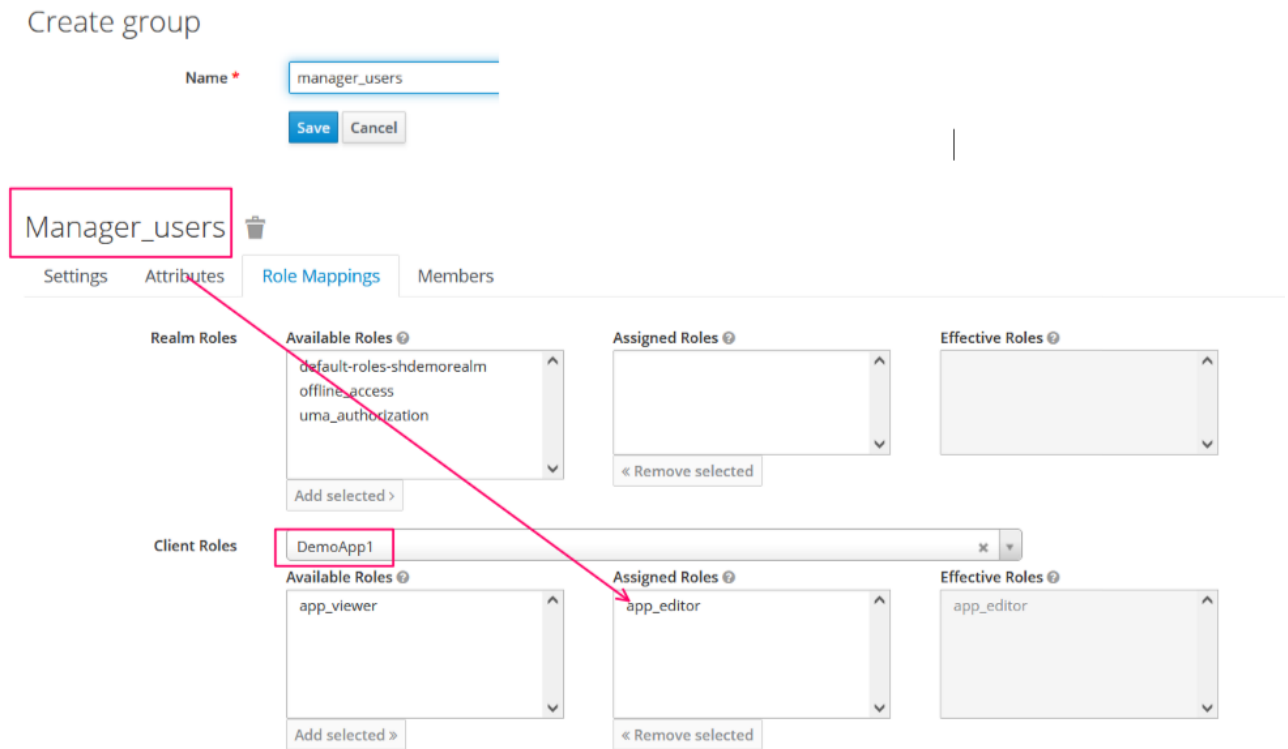
- **assistent_users** - група користувачів, що повинна тільки читати дані. Тобто цій групі повинна відповідати прикладна роль **app_viewer**.

- **manager_users** - група користувачів, що можуть змінювати дані. Тобто цій групі повинна відповідати роль **app_editor**.

Процес створення груп показано на [pic-12](#) та [pic-13](#).

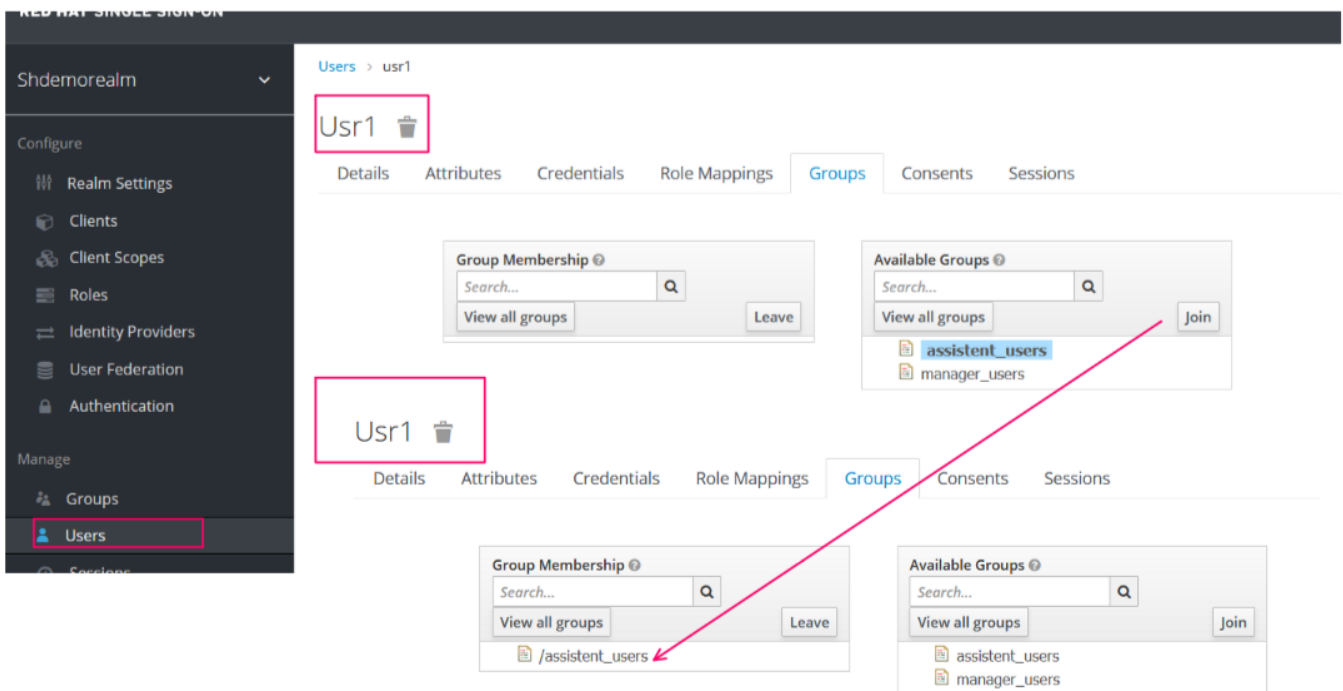


pic-12




pic-13

Далі, потрібно пройти по користувачах і додати кожного у відповідін групи так, як показано на [pic-14](#).



pic-14


Тут треба зазначити, що якщо keycloak інтегрувати з Active Directory - то групи користувачів будуть зразу відображатися і розносити користувачів по групах в KeyCloak не потрібно. Це робиться в Active Directory. У підсумку, користувачі по групах рознесені так, як показано на [pic-15](#).

Assistent_users 

Settings Attributes Role Mappings **Members**

| Username | Last Name | First Name |
|----------------------|------------|------------|
| usr1 | Петренко | Петро |
| usr2 | Семенченко | Семен |
| usr3 | Миколаєнко | Микола |

[Groups](#) > [manager_users](#)

Manager_users 

Settings Attributes Role Mappings **Members**

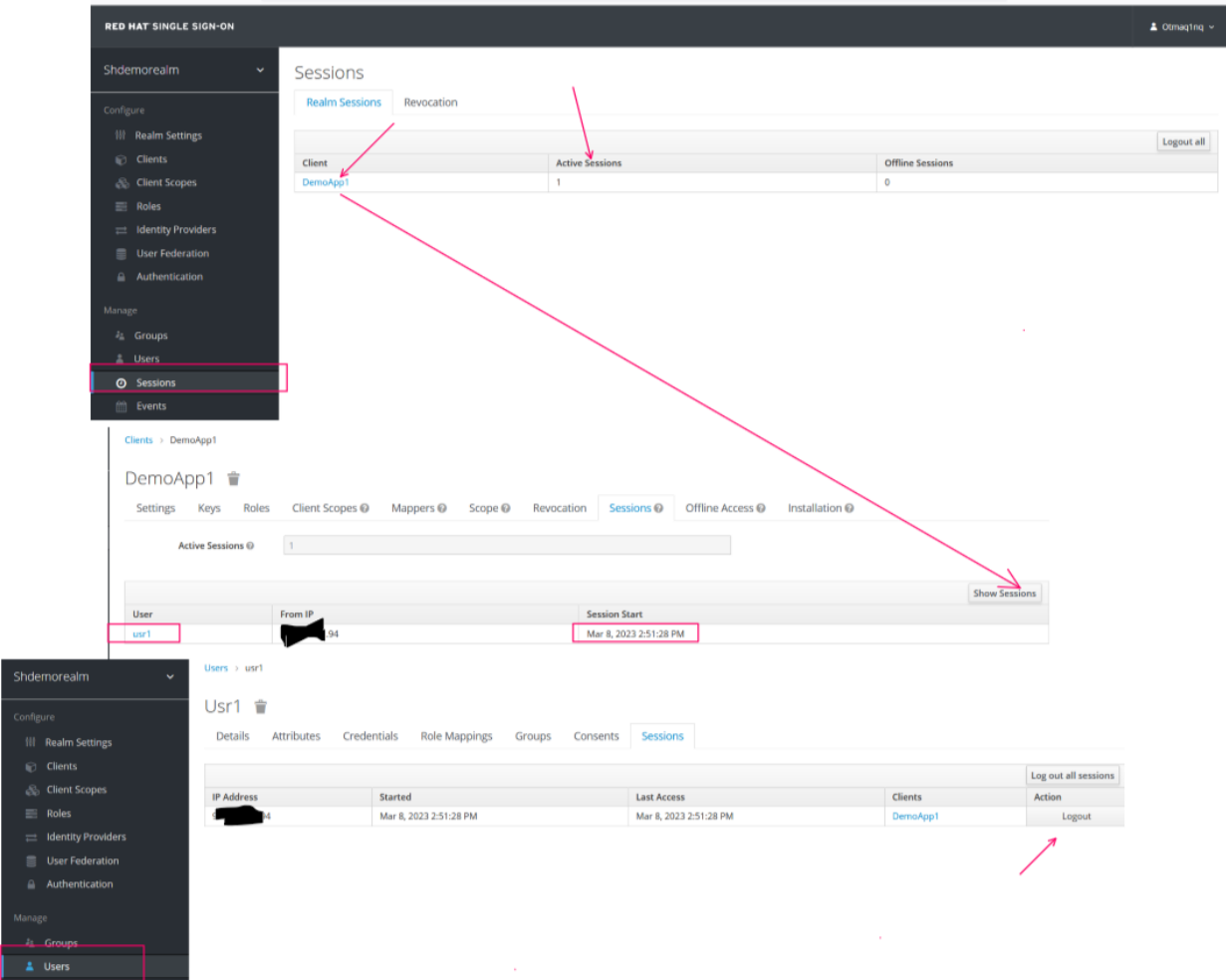
| Username | Last Name | First Name |
|----------------------|-----------|------------|
| usr4 | Денисов | Денис |

pic-15

На цьому етапі співставлення: користувачі-групи-client roles виконано. можна переходити до етапа тестування логіну та отримання авторизаційног токена.

Отримання авторизаційного токена по протоколу openid-connect

Для перевірки авторизації потрібно визначити, для початку, знайти "правильні" URL. Для цього. ідемо в налаштування realm та отримуємо json, як показано на [pic-16](#).



pic-17