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

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

Продук Keycloack 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.

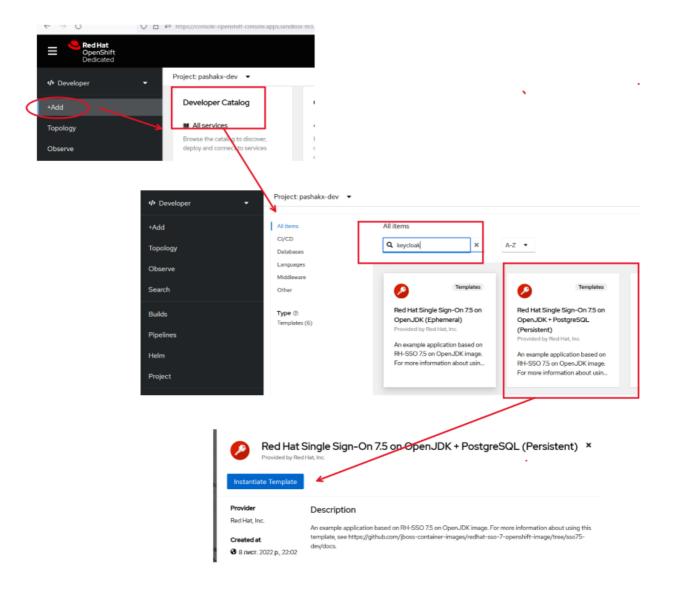
Ну на цьому зацікавився цим продуктом, щоб трохи розібратися як він працює та як його конфігурвати.

Розгортаня keycloak в OpenShift

За звичай keycloak можна підняти в контейнері, але він страртує в development mode. Так також там треба прив'язати базу даних типу postgresql. Я вирішив піти більш простим шляхом і роззоргнув його в хмарі RadHat в sendbox OpsnShift. Openshift sendbox можна створити за url:

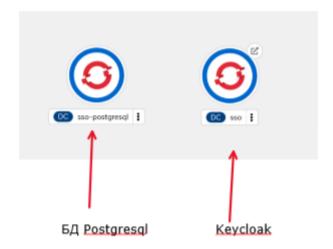
https://developers.redhat.com/developer-sandbox, а перед цим потрібно зарєструватися як developer на RedHat. Docker відкинув зразу, тому що прочитав ліцензійні обмеження для корпорацій.

Щоб не мучитися з лінкуванням бази даних та самого сервісу keycloak я використав калог уже підготованих продуктів, що є вже в любому OpenShift і в пару кліків розгорнув додаток pic-01.



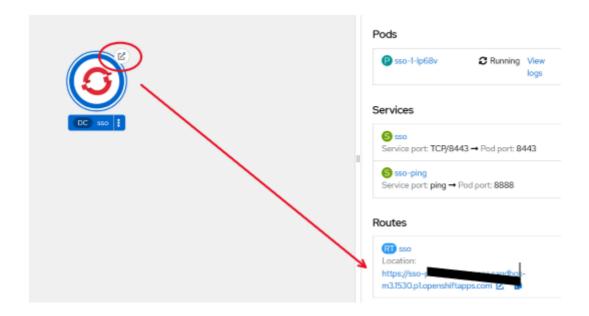
pic-01

Через кілька хвилин я уже маю готовий keycloak:



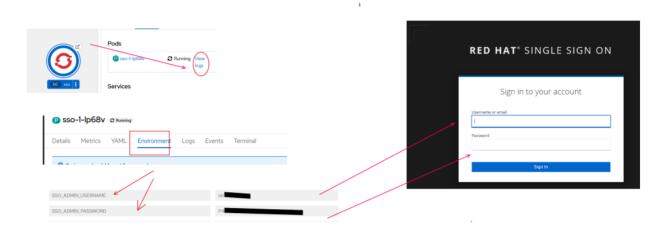
pic-02

I уже натискаємо на роут ріс-03, попадаємо в консоль адміністрування. Можливо потрібно трохи зачекати, поки обидва сервіса стартонуть. Ну, там час старту такий собі, відчутний, але не критично.



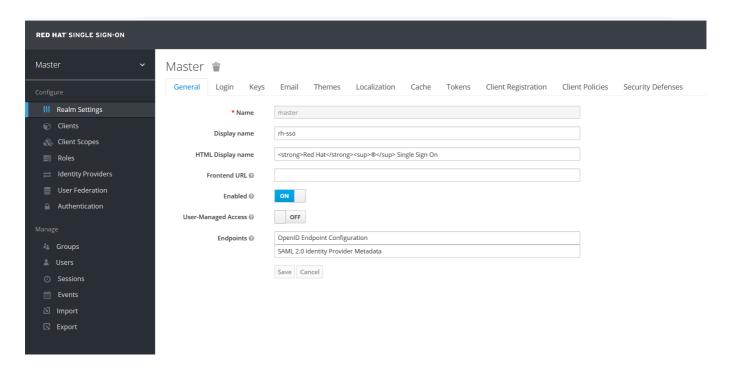
pic-03

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



pic-04

Залогінился і вуаля - попадаємо в консоль адмінітрування, в головний realm

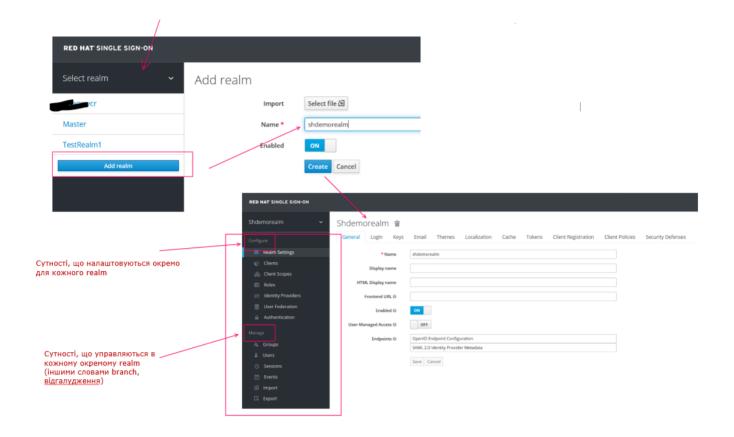


pic-05

Основні терміни адмінітрування Keycloak

Адміністрування в keycloak ділиться на realm-и.

• **Realm** це одниця адміністрування, що інкапсулює в собі набори: користувачів, ролей, груп та набори додаткових повноважень (чи прав). Тому, переше, що робимо, реєструємо свій realm. Назвемо його **shdemorealm**. На ріс-06 показані основні елементи realm.

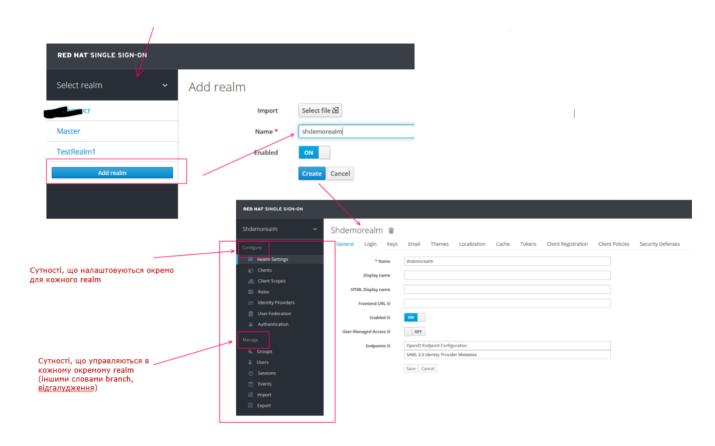


Keycloak підтримує як OpenID Connect (розширення OAuth 2.0), так і SAML 2.0. Коли говоримо про security, перше, що потрібно вирішити, це те, що з двох ви збираєтеся використовувати. Я вирішую використовувати OpenID Connect (розширення OAuth 2.0), тому в подальшому мова йде тільки про нього.

• Client - це прикладні додатки, що зареєстровані в окремо взятій realm. За звичай додатки аутентифікуються за допомогою clinet 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.

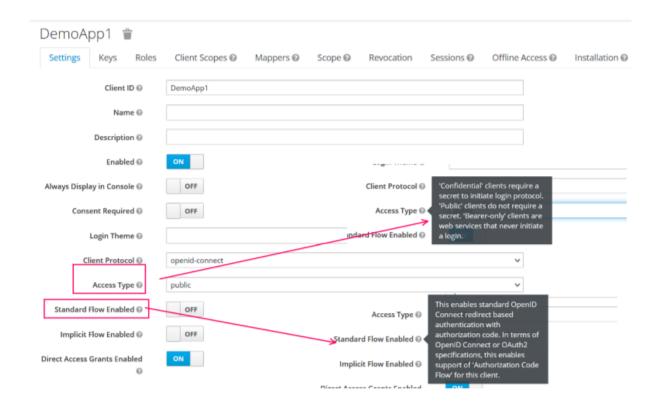
Ручна реєстрація програми клієнта

Створюємо клієнта в client ID **DemoApp1**, як показано на pic-06:



pic-06

Після реєстрації зразу попадаємо на вікно ріс-07

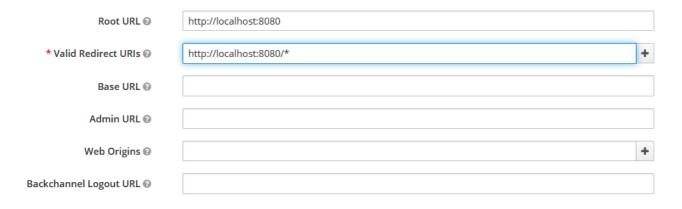


pic-07

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 autorization_code.

Далі потрібно налаштувати URL додатка так, як показано на ріс-08.



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

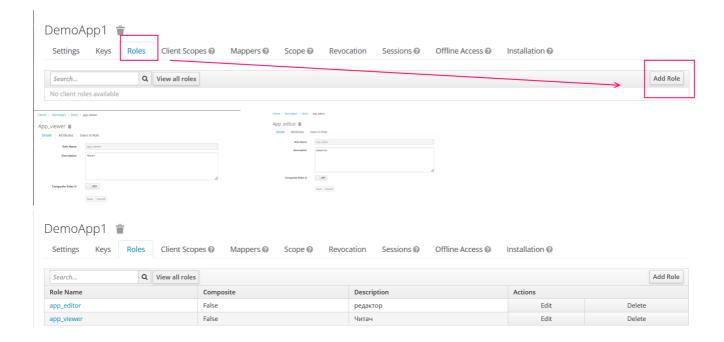
Після реєстрації додатку-клієнта потрібно запараметризувати рольовий доступ до ресурсів програми. Рольовий досутп може ділитися на по ролям. А ролі діляться на Realm Roles та Client Roles.

- 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 майбутньго додатку створемо 2 ролі:

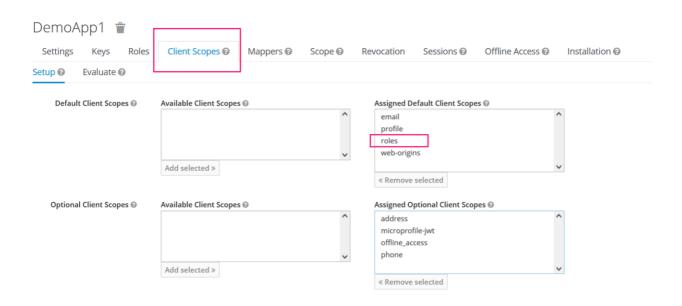
- app_viewer дозволяє тільки читати дані:
- app_editor дозволяє читати та змінювати дані.

Для цього створюємо ролі, як показано на ріс-09



pic-09

Щоб ролі передавалися на клієнта, потрібно пересідчитися, що вони включені в client scope, як на ріс-10.

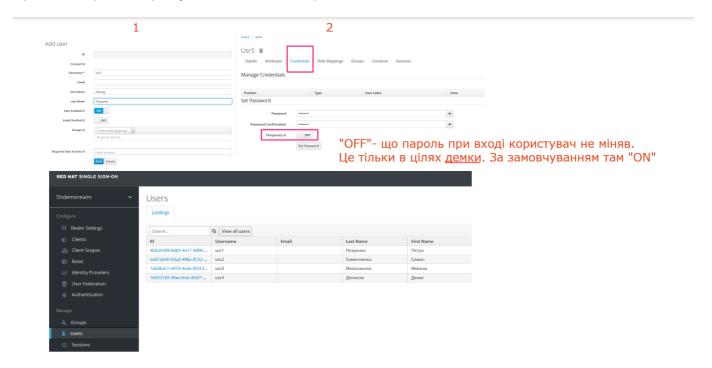


pic-10

Створення користувачів для призначення їм ролей

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

Процес створення користувачыв показано на ріс-11.



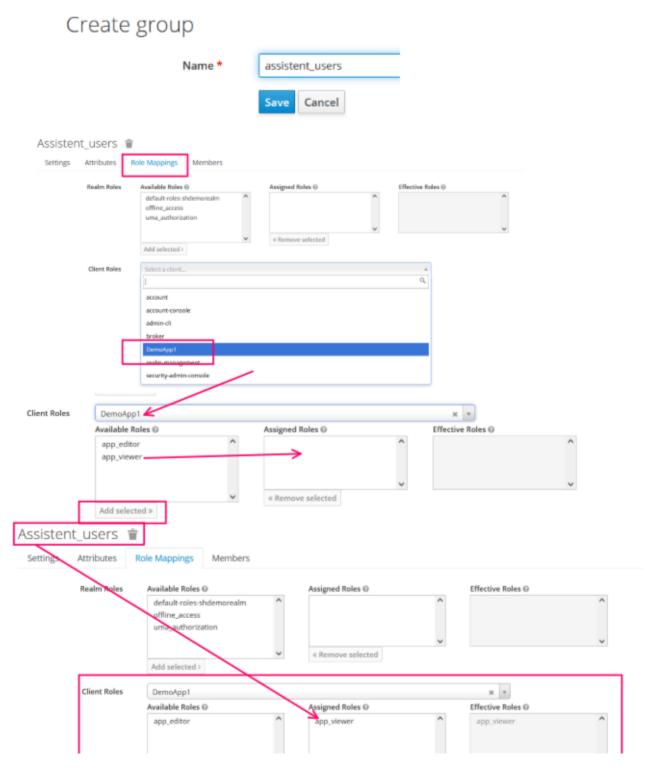
pic-11

Тепер створимо групи користовучів і поділимо користувачів на дві групи:

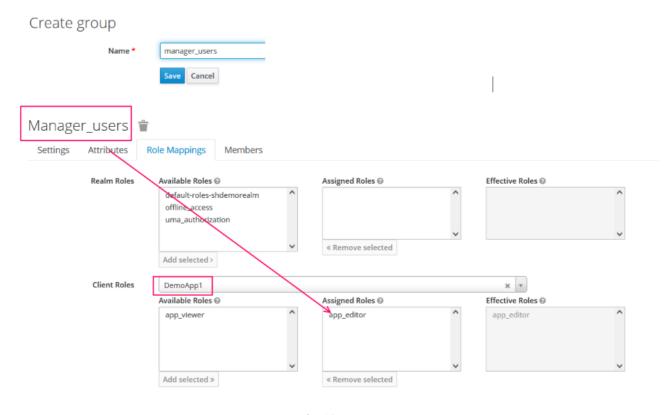
• assistent_users - група кристувачів, що повинна тільки читати дані. Тобто цій групі повинна відповідати прикладна роль app_viewer.

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

Процес створення груп показано на ріс-12 та ріс-13.

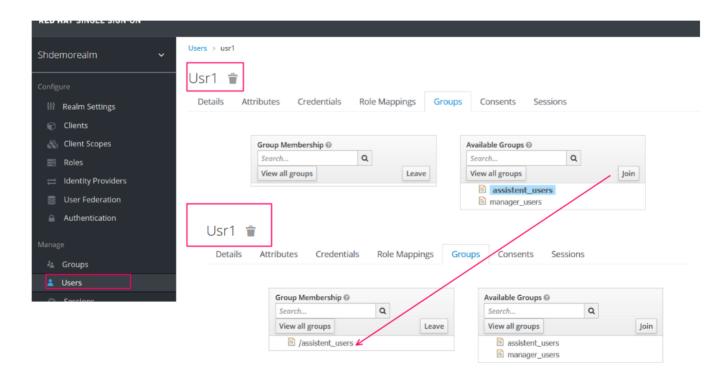


pic-12



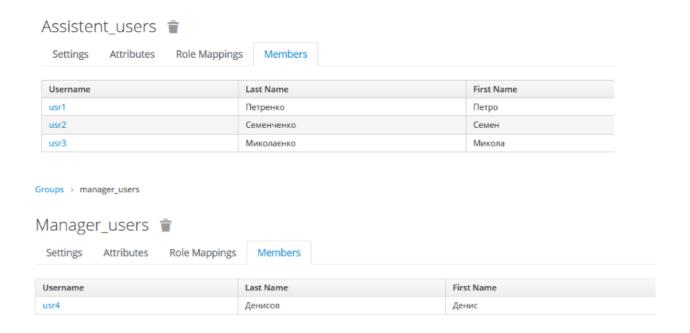
pic-13

Далі, потрібно пройтися по користувачах і додати кожного у відповідін групи так, як показано на ріс-14.



pic-14

Тут треба зазначити, що якщо keycloak інтегрувати з Active Directory - то групи користувачів будть 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.

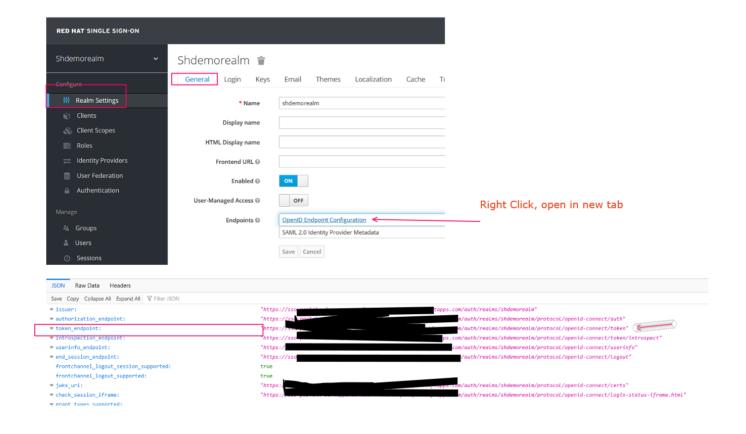


pic-15

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

Отримання авторизаційного токену 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-16

Далі беремо url в ключі token_endpoint та формуємо http - запит на KeyCloack

- Method: Http=POST
- URL \${token_endpoint}
- Http Headers

```
Content-Type: application/x-www-form-urlencoded
```

Request Body:

```
grant_type=password&client_id=DemoApp1&username=usr1&password=11111111
```

• Response Body:

```
{
    "access_token":
    "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJaUUhtZ0oya2R1b1FPOG12NTVoYU1vO
    XJKd1hMYVBGZ3VJRU9JejA5Y0xvIn0.eyJleHAiOjE2NzgyODAxODgsImlhdCI6MTY3ODI3OTg4OCwianR
    pIjoiMjE3NGNhNmUtZjcxMC00MDFkLWJiOGMtYzBjOTBlZWI0N2VhIiwiaXNzIjoiaHR0cHM6Ly9zc28tc
    GFzaGFreC1kZXYuYXBwcy5zYW5kYm94LW0zLjE1MzAucDEub3BlbnNoaWZ0YXBwcy5jb20vYXV0aC9yZWF
```

sbXMvc2hkZW1vcmVhbG0iLCJhdWQiOiJhY2NvdW50Iiwic3ViIjoiNGJiNTQzZDktOGQ4NS00ZTcxLTlkO
TYtMTA1YWNiNTE2MjNlIiwidHlwIjoiQmVhcmVyIiwiYXpwIjoiRGVtb0FwcDEiLCJzZXNzaW9uX3N0YXR
lIjoiMzBhN2YwODMtNGFjZi00ZWY2LTg0MGEtZDA4YTYzZmMyNzAyIiwiYWNyIjoiMSIsInJlYWxtX2FjY
2VzcyI6eyJyb2xlcyI6WyJkZWZhdWx0LXJvbGVzLXNoZGVtb3JlYWxtIiwib2ZmbGluZV9hY2Nlc3MiLCJ
1bWFfYXV0aG9yaXphdGlvbiJdfSwicmVzb3VyY2VfYWNjZXNzIjp7IkRlbW9BcHAxIjp7InJvbGVzIjpbI
mFwcF92aWV3ZXIiXX0sImFjY291bnQiOnsicm9sZXMiOlsibWFuYWdlLWFjY291bnQiLCJtYW5hZ2UtYWN
jb3VudC1saW5rcyIsInZpZXctcHJvZmlsZSJdfX0sInNjb3BlIjoicHJvZmlsZSBlbWFpbCIsInNpZCI6I
jMwYTdmMDgzLTRhY2YtNGVmNi04NDBhLWQwOGE2M2ZjMjcwMiIsImVtYWlsX3ZlcmlmaWVkIjpmYWxzZSw
ibmFtZSI6ItCf0LXRgtGA0L4g0J_QtdGC0YDQtdC90LrQviIsInByZWZlcnJlZF91c2VybmFtZSI6InVzc
jEiLCJnaXZlbl9uYW1lIjoi0J_QtdGC0YDQviIsImZhbWlseV9uYW1lIjoi0J_QtdGC0YDQtdC90LrQviJ
9.SvsPQPDIac3YwF7bVYFf3N1W4gkoYcIYppfTOVWKPAiQQLCt4WgFKGIGPf_OtbGikMLpm7cSHJAoc3IOOVWR

xsR0RTac3YwF7hVX5f3N1W4qkoXcIXnnfIQVMKRAi99LCt4UqEKGIGBf_QtbGikMLnm7sSHJAesJIOoVUR Yk1VNM0Hxb6xgy4gy6KGcgxabbeuDdLJsN2OVENTtFL9m3GQAAlN71w7ka8MmMxcxqYhEAxQtVwvCU1pa6 tpkOpoQKpRB9CeNRuhx9KdziymvNR4AEhD8Q1QqB0yLvyjxQ_pkKGT808IpWPezvqMGXuLQ2yEfDV8sc3PFyOMhDCwLj5W-

egvyJJq2bIgssNNYJ69WsX_WcnfPWOisErGIMijcNq7rFCEwrmIUdrTaTxyEfeJfJgU0MjJd1a-440WIA",

```
"expires_in": 300,
"refresh_expires_in": 1800,
"refresh token":
```

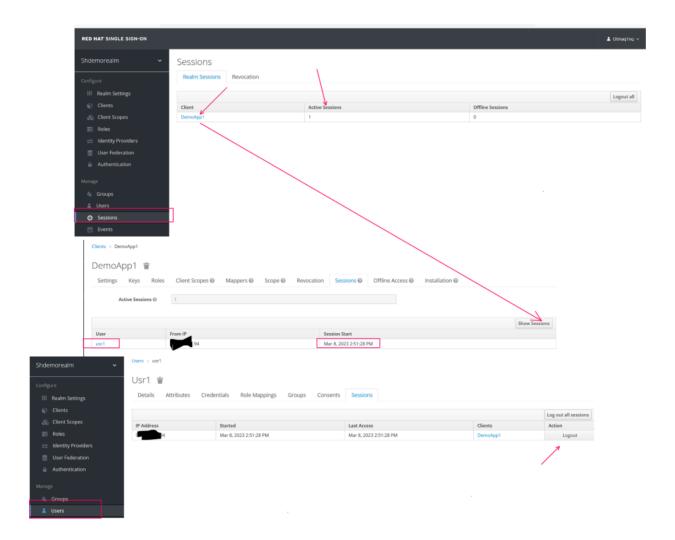
"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJkYTVmZWIwOS1hNWYxLTQwZjMtYjc5M C1iYTIONGJkYTIyMDIifQ.eyJleHAiOjE2NzgyODE2ODgsImlhdCI6MTY3ODI3OTg4OCwianRpIjoiNjkO MTRkMGYtMTAyZSOOYzc1LThlNmQtNjRmZmY1ZjhmODhjIiwiaXNzIjoiaHR0cHM6Ly9zc28tcGFzaGFreC 1kZXYuYXBwcy5zYW5kYm94LW0zLjE1MzAucDEub3BlbnNoaWZ0YXBwcy5jb2OvYXVOaC9yZWFsbXMvc2hk ZW1vcmVhbG0iLCJhdWQiOiJodHRwczovL3Nzby1wYXNoYWt4LWRldi5hcHBzLnNhbmRib3gtbTMuMTUzMC 5wMS5vcGVuc2hpZnRhcHBzLmNvbS9hdXRoL3JlYWxtcy9zaGRlbW9yZWFsbSIsInN1YiI6IjRiYjU0M2Q5 LThkODUtNGU3MSO5ZDk2LTEwNWFjYjUxNjIzZSIsInR5cCI6IlJlZnJlc2giLCJhenAiOiJEZW1vQXBwMS IsInNlc3Npb25fc3RhdGUiOiIzMGE3ZjA4My0OYWNmLTR1ZjYtODQwYS1kMDhhNjNmYzI3MDIiLCJzY29w ZSI6InByb2ZpbGUgZW1haWwiLCJzaWQiOiIzMGE3ZjA4My0OYWNmLTR1ZjYtODQwYS1kMDhhNjNmYzI3MD IifQ.m6MbJ1bVlbixDzURut5vOPmWVS5aD2o2GCw3v2OYOYO",

```
"token_type": "Bearer",
  "not-before-policy": 0,
  "session_state": "30a7f083-4acf-4ef6-840a-d08a63fc2702",
  "scope": "profile email"
}
```

Hy, або у вигляді curl

```
curl -X POST -k -H 'Content-Type: application/x-www-form-urlencoded' -i
'https://[domain.name]/auth/realms/shdemorealm/protocol/openid-connect/token' --
data 'grant_type=password&client_id=DemoApp1&username=usr1&password=11111111'
```

Ну а якщо зайти в меню **sessions** або **users**, то можна побачити всі підключені сесії жо жаного клієнта, та є можливісь всіх, або окремо взятого користувача - відключити ріс-17.



pic-17