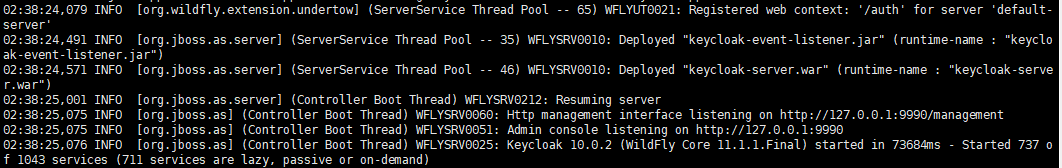
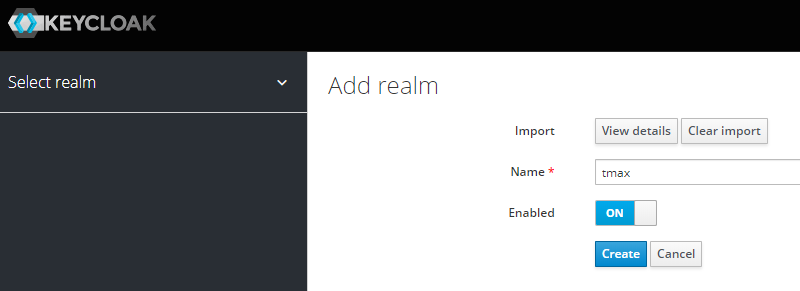
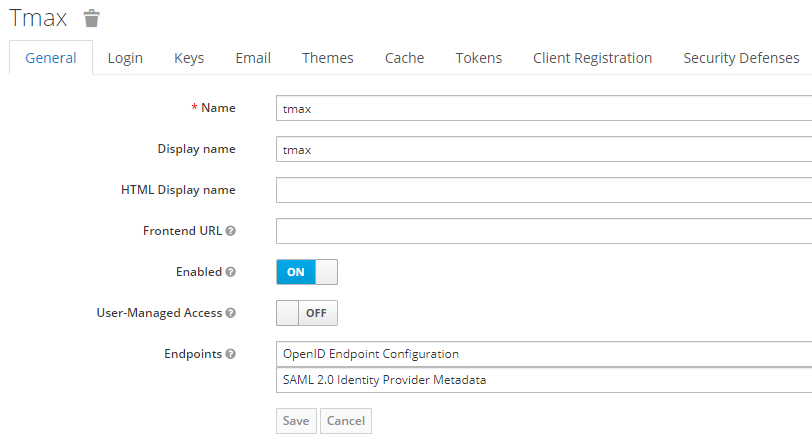
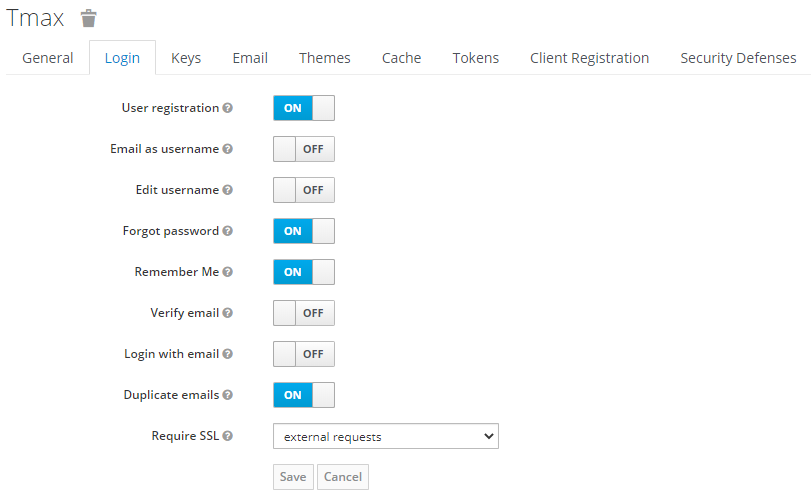
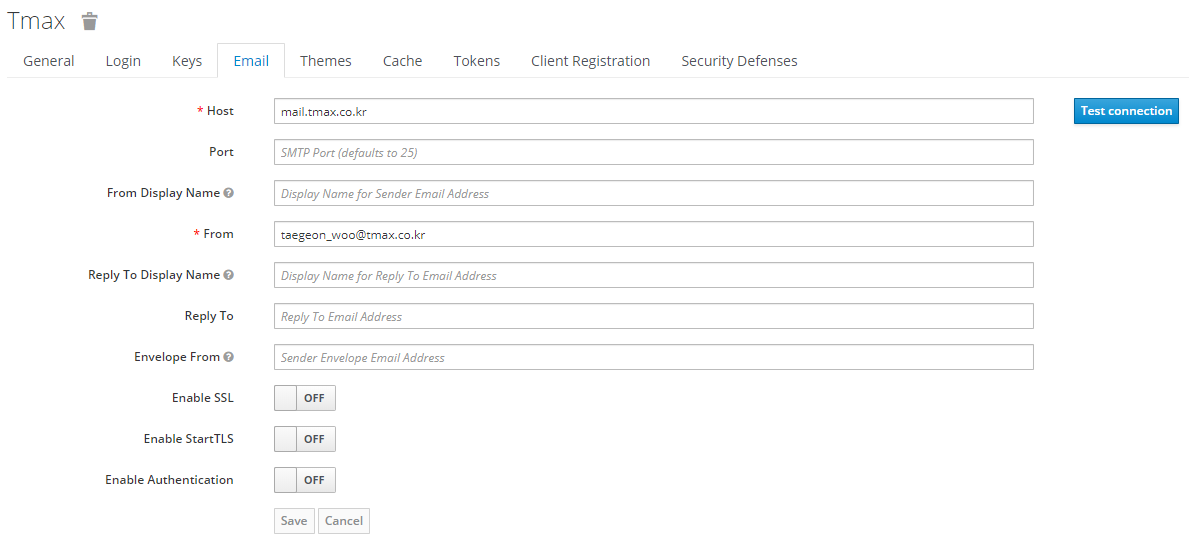
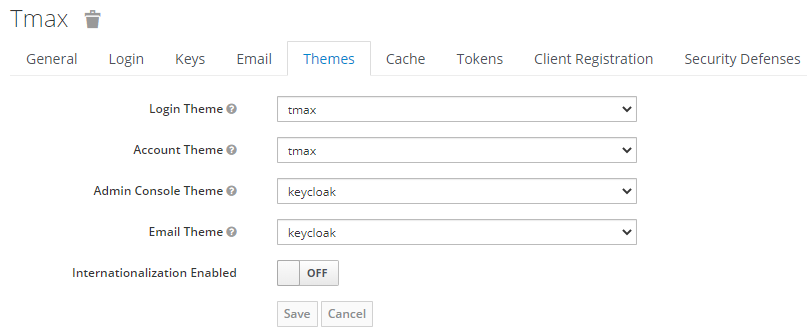
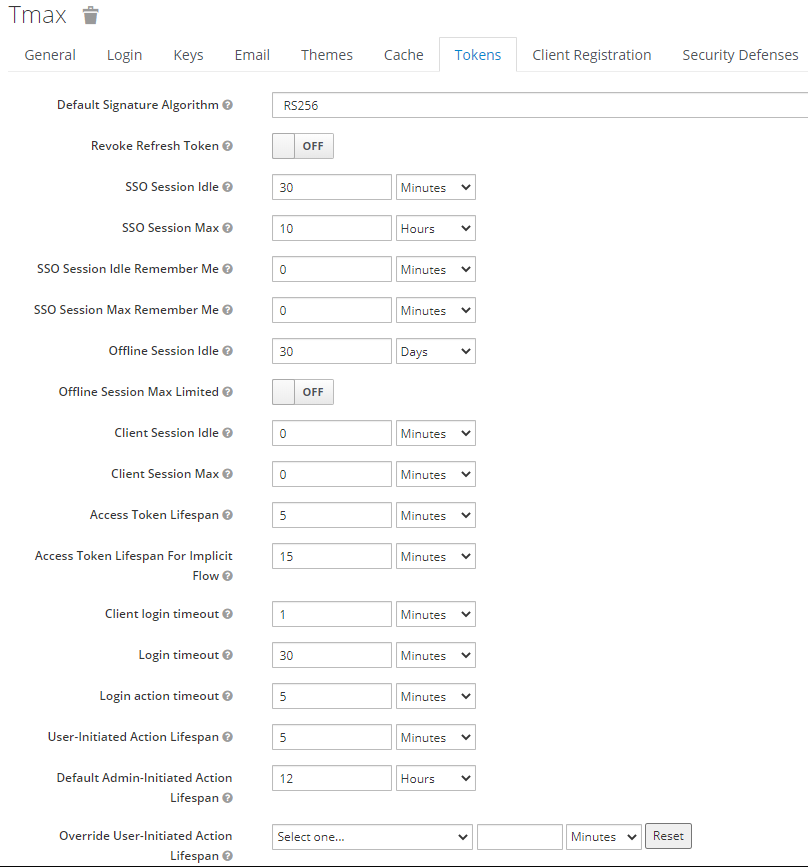
HyperAuth Admin Guide

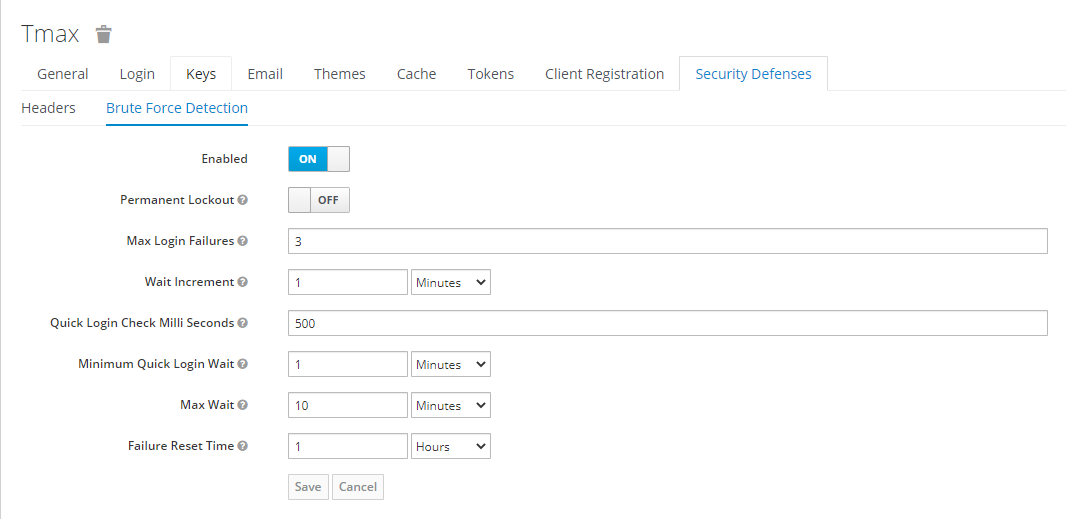
1. Install 가이드
   1. Namespace, ResourceQuota, ServiceAccount 생성
      1. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/init\_hyper\_auth.yaml 실행
   2. DB 생성
      1. Postgresql 의 경우
         1. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/postgre\_pvc.yaml 실행
         2. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/postgre\_service.yaml 실행
         3. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/postgre\_deploy.yaml 실행
         4. Postgresql 파드 기동 확인
            1. kubectl get pod -n hyperauth 명령어를 통해 뜨고 있는 postgresql pod 이름을 복사한다. Ex) postgresql-78d7b6c757-v2swm
            2. Ex) kubectl exec -it postgresql-78d7b6c757-v2swm -n hyperauth bash 명령어를 통해 postgresql pod로 접속한다.
            3. psql -U keycloak keycloak 명령어 입력으로 postgresql admin 접속 확인
      2. Tibero 의 경우
         1. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/tibero\_pvc.yaml 실행
         2. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/tibero\_standalone.yaml 실행
         3. Tibero 파드 기동 확인
            1. proauth-db는 최초로 기동시 20~30분 정도가 소요될 수 있음
            2. kubectl get pod -n hyperauth 명령어를 통해 뜨고 있는 tibero pod 이름을 복사한다. Ex) tibero-7bcbf5dc5b-gllhp
            3. Ex) kubectl exec -it tibero-7bcbf5dc5b-gllhp -n hyperauth bash 명령어를 통해 tibero pod로 접속한다
            4. /tibero 경로에 있는 cmdlog를 확인한다
            5. SQL> SQL> 2 3 4 5 6 7 8 9 10 11 12 SQL> SQL> Disconnected. 등으로 끝나는 default table을 만들어주는 로그가 있으면 완료가 된것
   3. HyperAuth 서비스 생성
      1. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/keycloak\_svc.yaml 실행
         1. ClusterIP type의 경우 nodePort 부분을 지워준다. (ex. nodePort: 32570, nodePort: 32573 삭제)
         2. kubectl get svc hyperauth -n hyperauth 명령어 입력을 통해, 노출하고자 하는 clusterIP 혹은 externalIP 확인 및 복사 (**{HYPERAUTH\_SERVICE\_IP}**)
   4. https를 위한 openssl 인증서 발급
      1. openssl req -newkey rsa:4096 -nodes -sha256 -keyout hyperauth.key -x509 -subj "/C=KR/ST=Seoul/O=tmax/CN={HYPERAUTH\_SERVICE\_IP}" -days 365 -config <(cat /etc/ssl/openssl.cnf <(printf "[v3\_ca]\nsubjectAltName=IP:{HYPERAUTH\_SERVICE\_IP}")) -out hyperauth.crt 명령어 입력
      2. 명령어를 입력한 창에 hyperauth.crt, hyperauth.key 생성 확인
      3. kubectl create secret tls hyperauth-https-secret --cert=./hyperauth.crt --key=./hyperauth.key -n hyperauth 입력해서 인증서 관리를 위한 secret 생성
      4. cp hyperauth.crt /etc/kubernetes/pki/hyperauth.crt 입력해서 hyperauth.crt 인증서 복사하기 (해당 경로에 가져다 놓은 hyperauth.crt 설정은 후에 kubernetes 연동가이드의 kube-apiserver.yaml 설정에 추가한다.)
   5. HyperAuth Deployment 생성
      1. http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install\_yaml/keycloak\_deploy.yaml 실행
         1. DB\_VENDOR, DB\_PORT 환경변수를 Tibero, Postgresql의 경우에 따라 설정한다.
         2. 약 5분 정도 소요
      2. Hyperauth파드 기동 확인
         1. Kubectl get pod -n hyperauth 명령어를 통해 뜨고 있는 hyperauth pod 이름을 복사한다. Ex) hyperauth-99f7dcdf6-wmgf6
         2. Ex) Kubectl logs -f hyperauth-99f7dcdf6-wmgf6 -n hyperauth 명령어를 통해 Hyperauth pod 의 기동 로그를 확인한다.
         3. 다음과 같은 로그가 보이면 정상적으로 기동
   6. Hyperauth Admin Console에 접속
      1. admin / admin으로 접속
   7. Kubernetes 연동 (Optional)
      1. Tmax Realm 생성 및 Import (2-a) Realm Import 참고)
      2. Clients - Create 클릭 - Client ID : hypercloud4 입력 및 생성 (Import에 포함)
      3. 연동하려는 kube master node에 접속
      4. vi /etc/kubernetes/manifests/kube-apiserver.yaml 를 통해 kube-apiserver 설정 변경
         1. --oidc-issuer-url=https://{HYPERAUTH\_SERVICE\_IP}/auth/realms/tmax 추가
         2. --oidc-client-id=hypercloud4 추가
         3. --oidc-username-claim=preferred\_username 추가
         4. --oidc-username-prefix=- 추가
         5. --oidc-groups-claim=group 추가
         6. --oidc-ca-file=/etc/kubernetes/pki/hyperauth.crt 추가
         7. --authentication-token-webhook-config-file=/etc/kubernetes/pki/webhook-config 주석처리 or 삭제 (기존에 Webhook을 통한 인증을 위해 존재 하던 설정)
         8. 자동으로 kube-apiserver 가 재기동됨
      5. keycloak에서 발급 받은 ID\_TOKEN을 이용해 k8s api call 가능함을 확인
         * 1. k8s cluster 내에서 ID\_TOKEN 발급받기
           2. curl 'https://**{HYPERAUTH\_SERVICE\_IP}**/auth/realms/tmax/protocol/openid-connect/token' \ -H 'content-type: application/x-www-form-urlencoded' \ --data 'grant\_type=password&username=admin-tmax.co.kr&password=admin&client\_id=hypercloud4&scope=openid' \ --compressed \ --insecure | grep id\_token
           3. id\_token 부분 복사 ({ID\_TOKEN})
           4. kubectl config view | grep server 입력 후, IP 부분 복사 (**{K8S\_API\_SERVER\_IP:PORT}**) ex) 172.22.6.2:6443
           5. curl 'https://**{K8S\_API\_SERVER\_IP:PORT}**/api/v1/services?limit=250' -H 'authority: **{K8S\_API\_SERVER\_IP:PORT}**' -H 'authorization:Bearer **{ID\_TOKEN}**' --compressed --insecure 명령어 call을 통해, Service 목록을 잘 받아 오는지 확인
           6. Keycloak client 등록을 통해 발급 받은 token을 통한 k8s API call 이 가능하다는 것을 의미한다.
2. Realm
   1. Realm Import
      1. 기본 Realm, Client, Role Setting이 들어간 Import 파일을 제공한다.
         1. Realm : tmax, tmax theme 설정 포함
         2. Client : hypercloud4, kiali 포함
         3. Role : hyperauth tmax realm all role 포함
         4. Events : Login Events Settings, Admin Events Settings 설정, hyperauth\_event\_listener 추가 포함
      2. Add realm을 클릭하고, Import - Select file을 클릭하여, <http://192.168.1.150:10080/hypercloud/keycloak-distribution/blob/master/install_yaml/tmax-realm-export.json> 를 추가한다.



* + 1. Client 의 URL을 환경에 맞게 변경해준다. ex) Root URL, Valid Redirect URIs, Base URL, Admin URL, WebOrigins
  1. Realm 설정
     1. General 
        1. Enabled : On 설정을 통해 tmax realm을 활성화한다.
        2. General 화면에서는 다른 것은 건드리지 않는다.
     2. Login : Client 별 로그인 화면을 구성할 수 있는 설정
        1. User registration : 로그인 화면에서 유저 등록을 할 수 있도록 한다.
        2. Email as username : email을 username으로 사용 하도록 한다.
        3. Edit username : user detail 페이지에서 username을 수정 할 수 있도록 허용한다. 정책상, username을 user Id로 사용할 것이기 때문에, Enabled 하지 않는다.
        4. Forgot password : 로그인 화면에서 Password를 잊어버렸을 경우, 재설정 할수 있는 페이지가 나올수 있도록 한다.
        5. Remember Me : keycloak이 session 영역 기반으로 로그인 정보를 저장하고, session은 browser 당 1개씩 생성되어서 공유하기 때문에  
           remember me 기능은 'browser를 닫았을 때도 로그인 상태를 유지하는 기능
        6. Verify email : 첫번째 로그인이나, 이메일 주소를 변경했을때, 이메일을 인증 하도록 하는 설정
        7. Login with email : 이메일로 로그인 할 수 있도록 하는 설정
        8. Duplicate emails : 유저별 이메일 중복 체크를 허용하는 설정
        9. Require SSL
           1. All requests : 모든 Client IP Address 에 대해서 https 가 필요함
           2. External requests : Localhost 나 private IP로는 https 없이 접근 가능
           3. None : 모든 Client IP Address로 부터 https없이 접근 가능
     3. Email
        1. Keyclaok 으로 부터 메일을 받기 위해서 Email 설정이 필요함
        2. 오른쪽 상단의 Admin 계정의 Email과 이름을 추가한다.
        3. Host : mail.tmax.co.kr로 입력
        4. From : 관리자의 이메일을 입력한다
        5. Test Connection을 클릭하여 Admin 계정의 Email로 Test Email이 오는것을 확인한다.
     4. Theme



* + - 1. HyperAuth 배경을 정할 수 있는 설정
      2. Tmax Theme으로 변경가능
    1. Tokens
       1. Default Signature Algorithm : RS256으로 고정한다. 다른 것으로 바꾸면, kubernetes가 신뢰하지 않는다.
       2. Access Token Lifespan : tmax realm내에서 client에게 발급되는 accessToken의 exp 만료시간
    2. Security Defenses
       1. 로그인 시 유저가 비밀번호를 반복적으로 잘못 입력 할시, 유저를 Lock Out할 수 있다.
       2. Security Defenses - Brute Force Detection 에서 설정 할 수 있다.



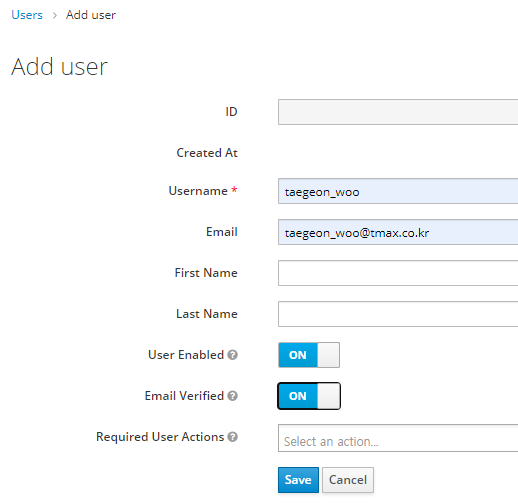
* + - * 1. Permanent Lockout

ON : 유저가 Max Login Failures를 초과할 시, 영원히 Lockout 하는 기능, 관리자가 Unlock 해주어야한다.

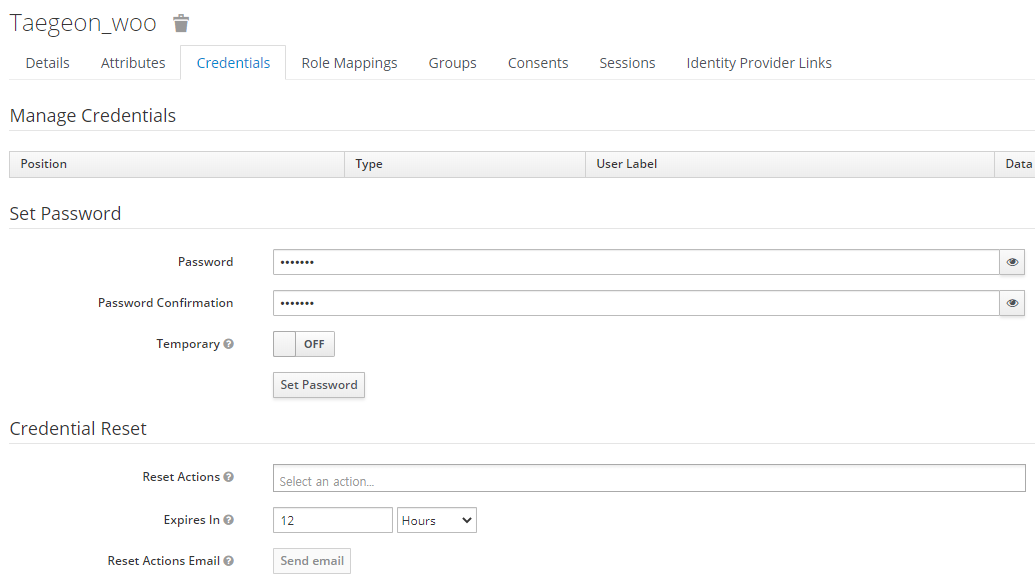
OFF : 유저가 Max Login Failures 를 초과할 시, 일시적으로 ( 정해진 시간 동안 ) Lockout 하는 기능

* + - * 1. Max Login Failures : Lockout되기 전 로그인 실패 최대 횟수
        2. Wait Increment : Wait 이 늘어는 기간
        3. Quick Login Check Milli Seconds : 이 기간 보다 빠르게 로그인을 연속적으로 실패하면, 유저는 Lockout 된다.
        4. Minimum Quick Login Wait : 빠르게 로그인 해서 Lockout 된 경우 Lockout 되는 기간
        5. Max Wait : 유저가 기다릴 수 있는 최대 기간
        6. Failure Reset Time : Login Failure Count를 초기화 해주는 데 걸리는 기

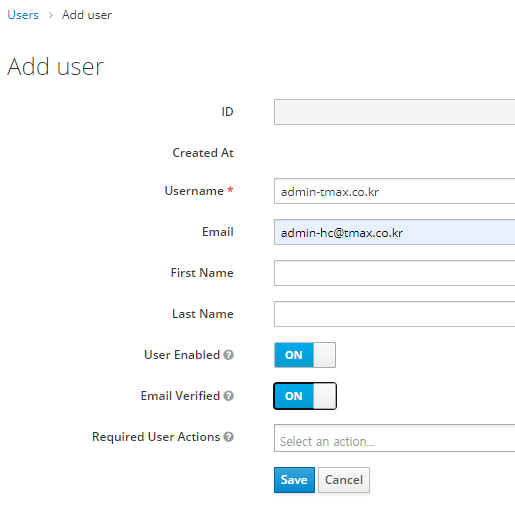
1. Users 관리
   1. User Add
      1. Add user 클릭

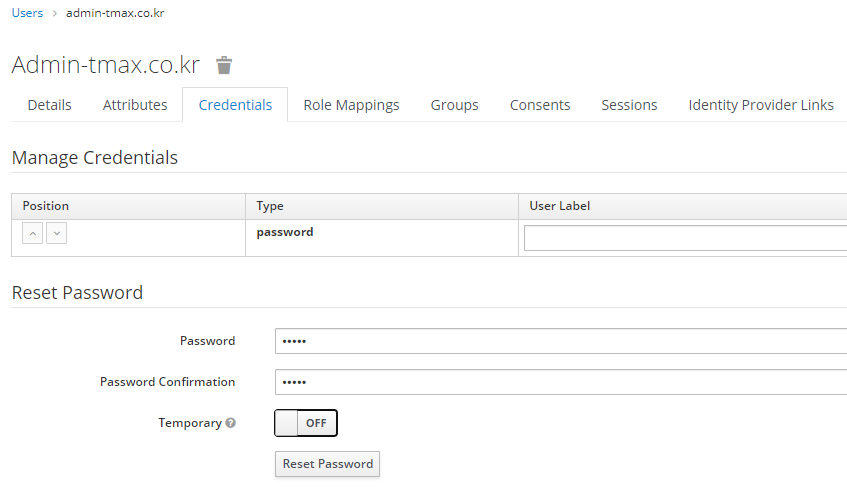


* + 1. Username은 userId로 사용할 예정 (Unique 해야한다)
    2. Email 을 입력하고 Email Verified를 ON으로 한다.
    3. Client 화면에서 로그인을 하기 위해서는 비밀번호 (Credentials)를 설정해야 한다. Temporary OFF로 바꾸고 설정한다.
    4. 마찬가지로, 비밀번호를 잊어버린 Client의 요청을 받아 admin이 Client 의 비밀번호를 대신 새롭게 reset 해줄 수도 있다.

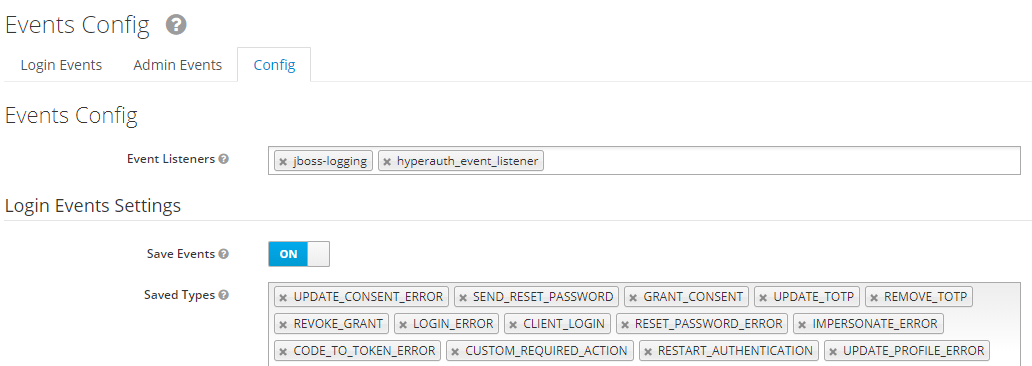


* + 1. 기본적으로 있어야 할admin-tmax.co.kr / admin 으로 유저를 생성한다. (Hypercloud admin 계정)

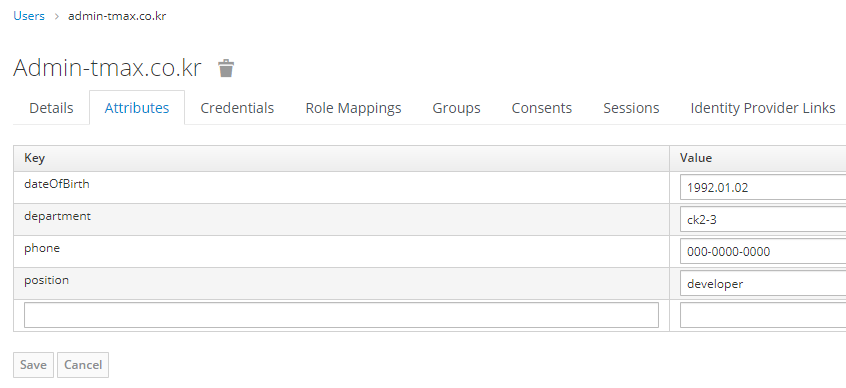




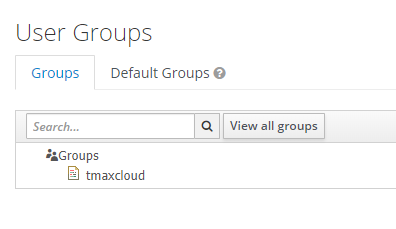
* 1. 참고로, HyperAuth 에서는 Client가 User Registration을 통해 유저가 새롭게 생성 될 경우, User가 Hypercloud4 환경에서 가져야 할 기본 Role, Role Binding이 만들어진다.
     1. 현재 기본 rule ( 20.08.20 기준)
        1. Namespace
           1. GET
        2. namespaceclaims
           1. CREATE, POST
        3. rolebindingclaims, resourcequotaclaims, catalogserviceclaims
           1. CREATE, GET, LIST
        4. clustermenupolicies
           1. GET
     2. Hyperauth\_event\_listener SPI를 통해 구현 되어 있음
        1. tmax realm을 Import했을 경우, Events Config에 hyperauth\_event\_listener 가 추가 되어 있음을 확인 할 수 있다.
        2. 새롭게 tmax realm을 추가 했거나, hyperauth\_event\_listener가 추가 되어 있지 않은 경우, 추가 해준다.



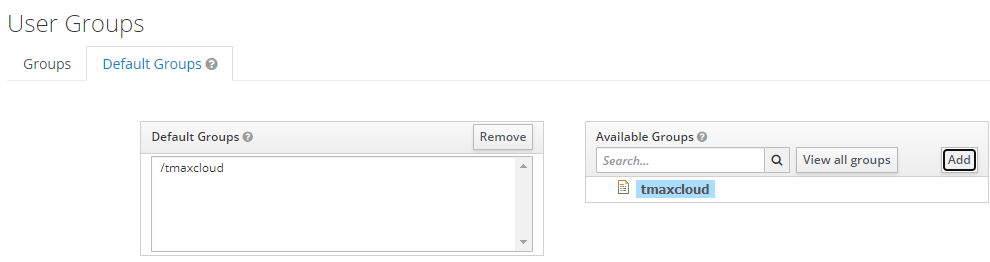
* + - 1. 설치에 쓰인 hyper-auth-server:b1.0.1.13 이미지에는 이미 hyperauth\_event\_listener가 추가 되어 있으므로, 손쉽게 추가 할 수 있다.
  1. User Migration 가이드
     1. Hypercloud v4.1.0 UserCRD Migration
        1. https://github.com/tmax-cloud/hypercloud-operator/blob/master/scripts/userMigrationFromK8s.sh 파일을 migration하려는 k8s환경에 위치시킨다.
        2. chmod +777 userMigrationFromK8s.sh 명령어 실행
        3. kubectl get svc -n hyperauth 명령어 실행을 통해, hyperauth 서비스의 http ip:port 를 복사한다. ex) 192.168.6.223:8080
        4. ./userMigrationFromK8s.sh 192.168.6.223:8080 으로 call 한다.
           1. 전제조건 : tmax realm은 만들어진 상태라고 가정한다.
           2. 이미 존재해서 중복되는 아이디가 있을 경우, 해당 유저를 skip 하고 migration이 된다.
        5. hyperauth admin console에서 tmax realm의 user가 추가된 것을 확인한다, attribute도 추가된 것을 확인한다. (phone, department, dateOfBirth, description 지원)
        6. 현재, UserId와 User Meta만 Migration된 상황이다. 비밀번호를 그대로 가져올 방법은 없기 때문에, Client가 비밀번호를 Reset해주어야 한다.
           1. client 로그인 화면에서 Forgot Password? 를 클릭한다
           2. 본인이 기존에 쓰던 userId를 Username에 입력한다.
           3. 등록되어 있던 메일로 전송 된 비밀번호 재설정 링크를 통해 비밀번호를 설정한다.
        7. UserMigration을 통해 가져온 User의 경우 앞서 언급했던, User의 기본 role은 생성 해 주지 않는다.
  2. User Detail
     1. Details : User의 기본설정을 바꿀 수 있다.
     2. Attributes



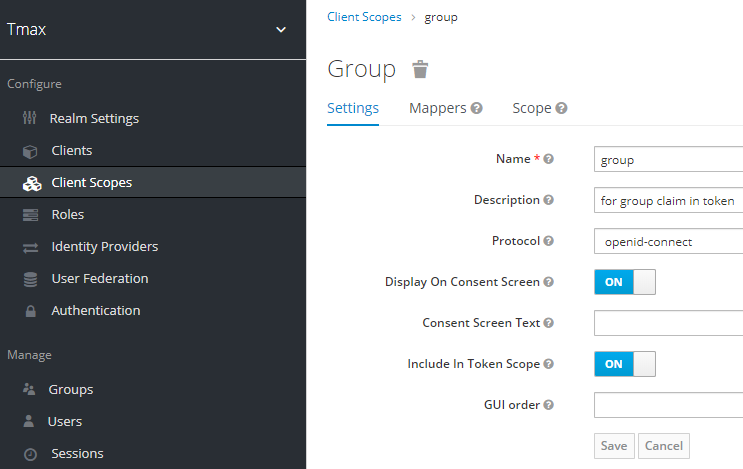
* + - 1. ID, Email을 제외한 User Meta를 Attribute으로 관리할 수 있다.
    1. Groups
       1. User Group 생성



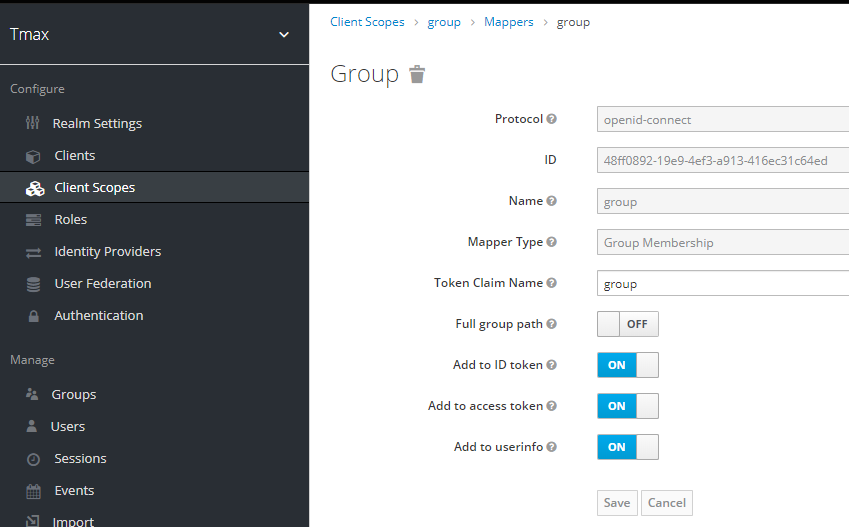
* + - * 1. Groups - New 클릭해서 UserGroup을 생성한다.
      1. Default Groups



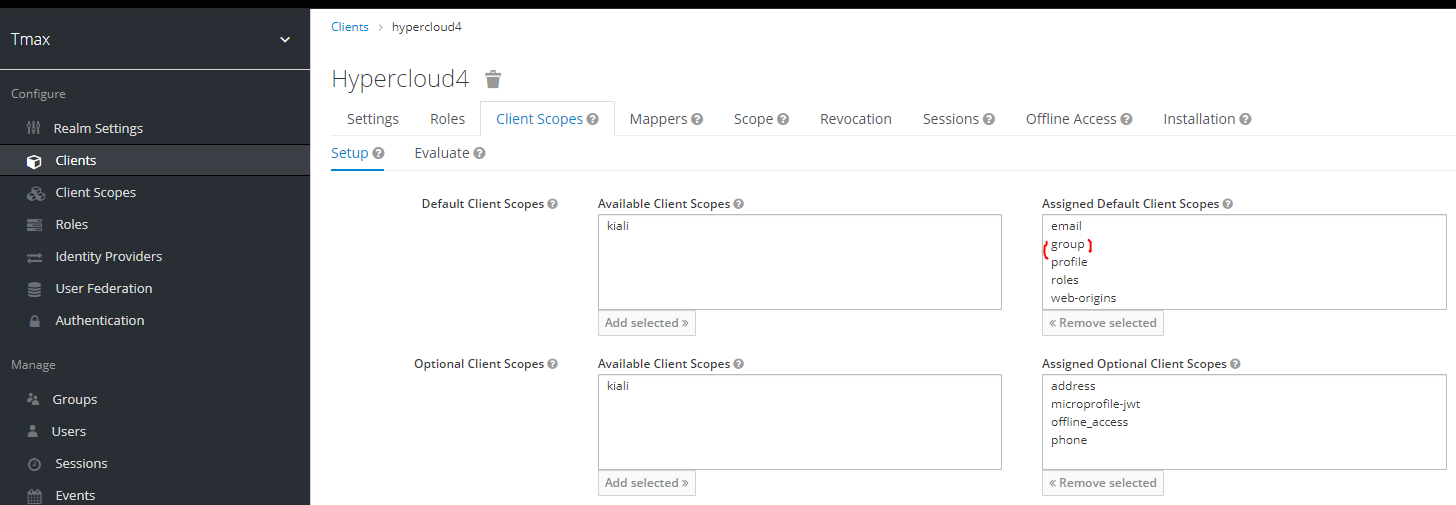
* + - * 1. User 생성시 자동으로 등록 될 UserGroup을 지정할 수 있다.
      1. User - admin-tmax.co.kr - Groups 에서 존재하는 그룹에 join 할 수 있다.
      2. ID 혹은 AccessToken에 Group Claim을 추가 하는 방법
         1. Client Scopes을 추가한다.



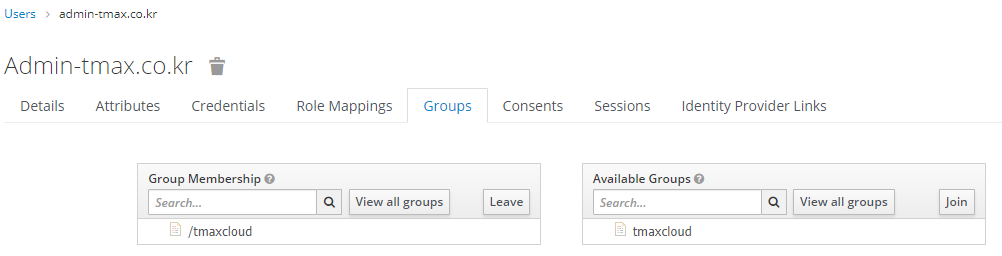
* + - * 1. ClientScopes의 Mapper에서 create을 누르고 group이라는 이름의 Mapper Type을 Group Membership, Token Claim Name을 group으로 해서 생성한다.



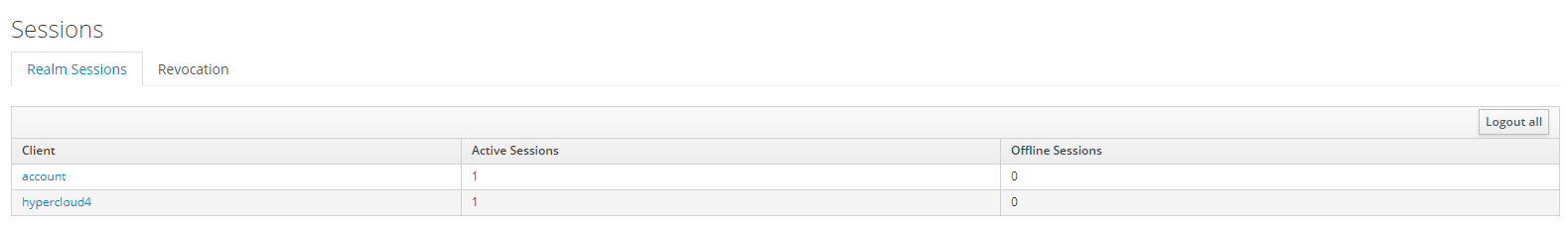
* + - * 1. Clients 별로 token 에 추가할지 말지 등록을 해주면 된다. Clients-Client Scropes에서 Default Client Scopes의 Avaliable Client Scopes에 생성된 group을 Add selected 해주면 된다.



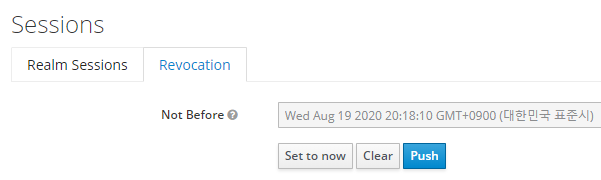
* + - * 1. 이제부터는 hypercloud4를 통해 로그인된 유저가 group에 속해 있다면, group이라는 claim에 group 정보가 들어가게 된다.
        2. 위의 k8s와의 연동과정에서 --oidc-groups-claim=group 을 추가 했기 때문에, k8s는 group이라는 claim 에 들어있는 정보를 group으로 써 신뢰 하게 된다.



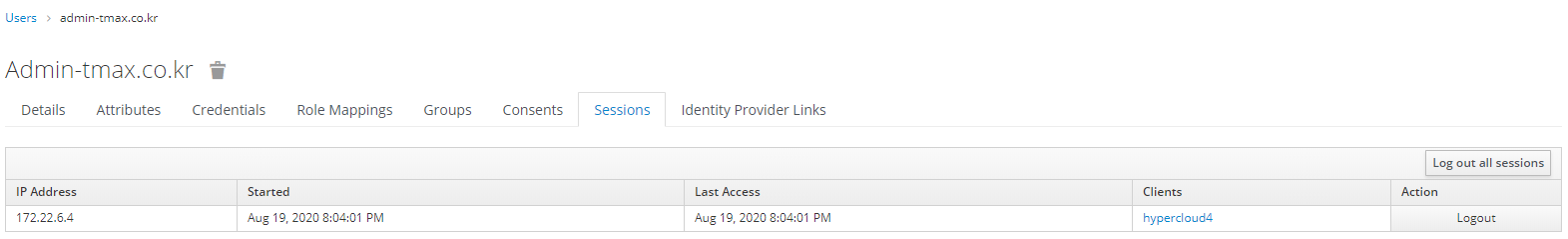
* + 1. Sessions



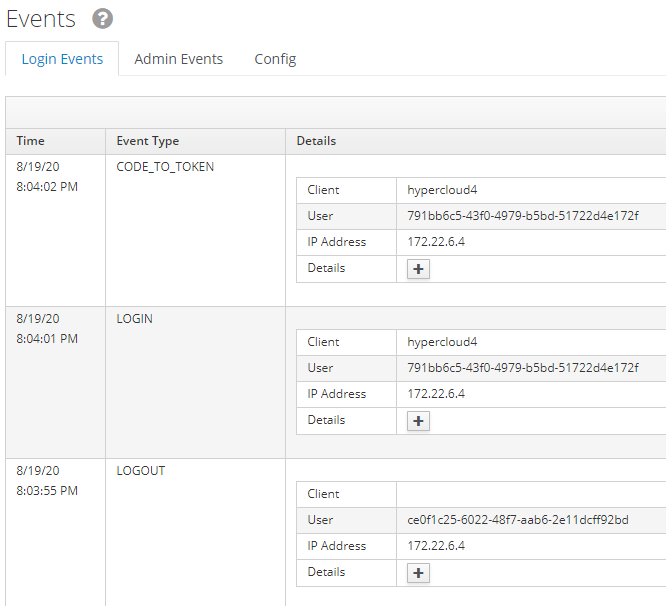
* + - 1. Realm 내에 로그인 되어있는 session전체를 Sessions 탭에서 볼 수 있다.
      2. Logout all을 통해 모든 Session을 종료 시킬 수 있다.



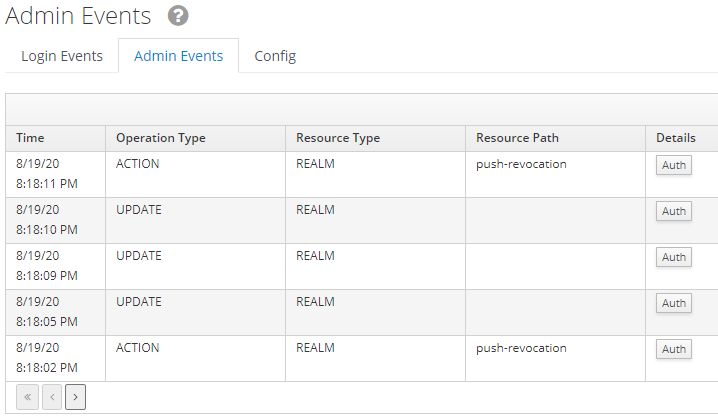
* + - 1. Sessions - Revocation을 통해 현재 시각을 기준으로 이전에 발행된 모든 Session 및 토큰을 적합하지 않게 만들 수 있다.



* + - 1. 유저기준으로 유저가 로그인 되어 있는 Client 및 Session 정보를 User - sessions 탭에서 얻을 수 있다
      2. 해당 유저를 로그아웃 시킬 수 있다.
    1. Events

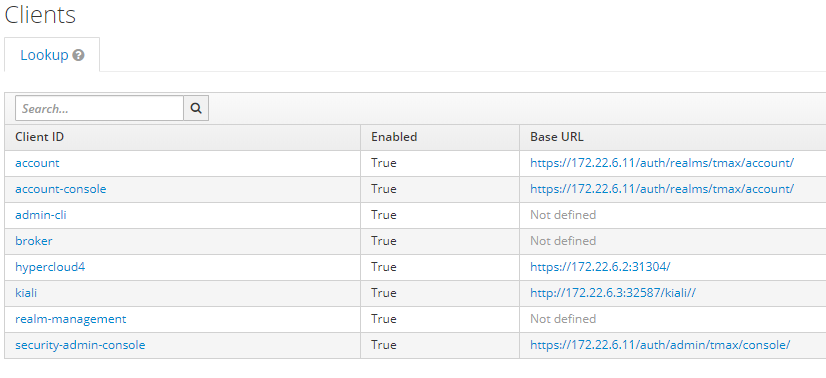


* + - 1. Login Events를 조회 할 수 있다.

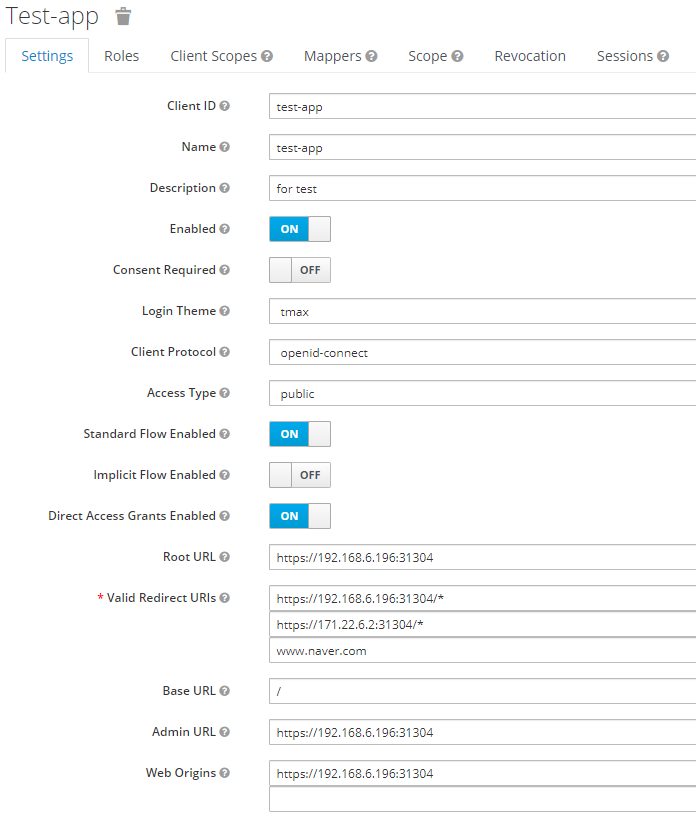


* + - 1. Admin Console에서 수행 된 Admin Events를 조회 할 수 있다.

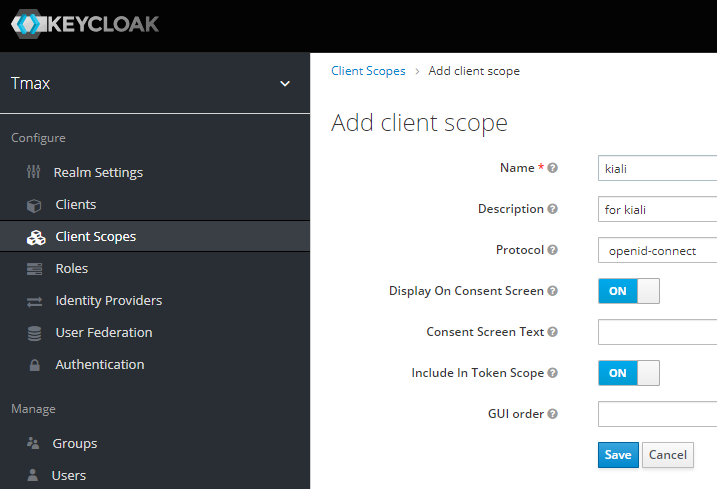
1. Client 관리
   1. Client 등록
      1. tmax realm을 Import 했으면, 기본적인 Client가 등록되어 있음을 볼 수 있다. Ex ) hypercloud4, kiali



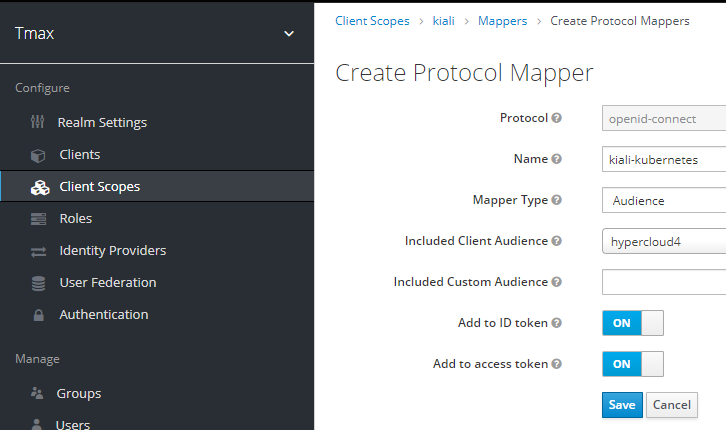
* + 1. Client를 추가 할 경우 Client ID에 Client Name을, Root URL 에는 Client의 도메인 시작 페이지 url을 입력한다.
    2. Client Settings

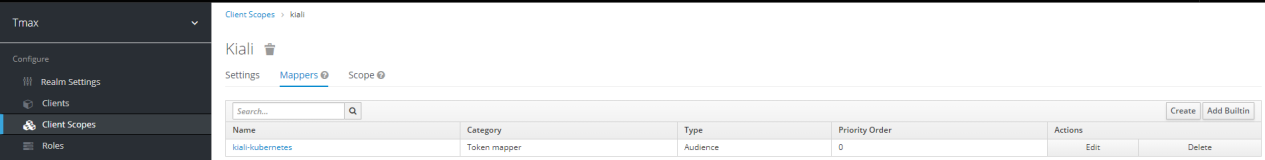


* + - 1. Enabled : Disable 하면 해당 Client를 통해 로그인을 할 수 없다. ON 으로 설정한다.
      2. Consent Required : OFF 로 둔다.
      3. Client Protocol : openid-connect로 둔다.
      4. Access Type : public 으로 둔다.
      5. Standard Flow Enabled : ON
      6. Implicit Flow Enabled : OFF
      7. Direct Access Grants Enabled : ON
      8. Valid Redirect URIs
         1. 로그인 혹은 로그아웃시 redirect 되게 할 uri를 적으면 됨
         2. 복수개 등록 할수 있다.
         3. \* 을 사용하여 wildcard로 동작 시킬 수 있다.
      9. Base URL : /을 설정하면 client 목록화면에 root url / 으로 노출시키고 링크가 형성 된다.
      10. Web Origins : 허용하는 CORS Origins
    1. Clients Settings - Advanced Settings
       1. Access Token Lifespan : Client별로 AccessToken의 Exp 시간을 조정할 수 있다.
  1. Kubernetes Call 이 필요한 Client의 경우
     1. 앞선 방식으로Client를 등록하면, Realm 내부에서 등록된 Client 끼리의 OIDC 및 SSO는 구현이 끝났다고 말할 수 있다.
     2. 이제 추가적으로, Client 중에서 내부적으로 Kubernetes Call을 해야하는 경우, 처리해주어야 할 추가 설정에 대해서 알아보자.
     3. 예를 들어, Prometheus, Kiali와 같은 경우, User가 Kiali Client로써, HyperAuth에 로그인 하는 경우, Hyperauth 에서 발급받은 토큰을 이용해서, Kiali가 내부적으로 Kubernetes Call을 할 수 있어야 Kiali가 정상적으로 동작한다고 말할 수 있다.
     4. 앞선 Kubernetes 연동 과정에서 kube-apiserver 설정에 --oidc-client-id=hypercloud4 다음과 같은 설정을 추가 했었다.
     5. 즉, kubernetes는 client id가 hypercloud4 인 경우만, 신뢰를 한다. 이를 위해, kiali를 통해 발급 받은 토큰의 Audience Claim에 hypercloud4를 추가 해주는 작업이 별도로 진행되어야 한다. kiali를 예를 들어 설명한다.
        1. Client Scopes를 클릭하여, kiali가 사용하게 될 Client Scopes를 만든다.

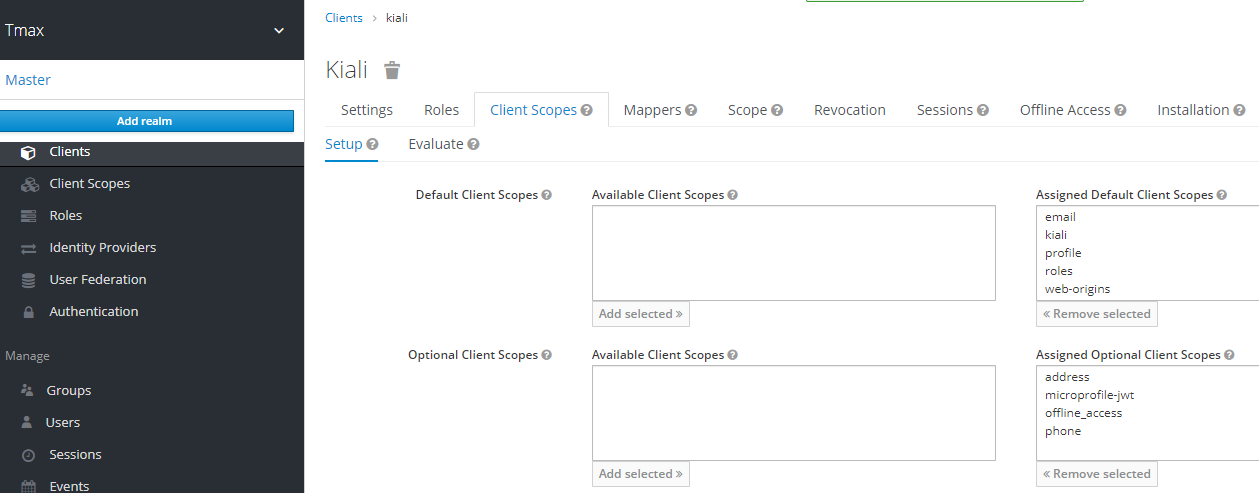


* + - 1. Kiali ClientScopes의 Mapper를 생성해준다. Mapper Type을 Audience로 해주고, Included Client Audience 에는 Client로 등록했던 hypercloud4를 추가해준다.





* + - 1. Clients - Kiali - ClientScopes 탭에 들어가서, Avaliable Client Scopes에 있는 kiali를 Add selected로 Assigned Default Client Scopes로 옮겨준다.



* + - 1. Kiali Client를 통해 토큰을 발급 받고, Token을 까보면, aud Claim에 kiali 뿐아니라, hypercloud4가 추가 되어 있음을 확인할 수 있다.

1. Identity Providers ( Social Login )
   1. 현재 지원 가능한 검증된 social login
      1. Github, Naver, Kakao
      2. 각 모듈의 developer 페이지에서 client ID, client Secret을 발급받아서