

- Introduction
 - Recommended way to study
 - Basic terminology
 - Architecture Overview
 - Components
 - Flow
- Local installation
 - Keycloak
 - Create a Realm and a Client
 - Create and configure the IDP
 - Important Notes
 - Nginx
 - Result
 - OpenShift installation
 - Upload to OpenShift using a Template
 - Extra Configuration
- Authenticate and Authorize
 - myIDP Client Registration
 - Configuring Authorization (blocking users)
 - Checking its actually working

Introduction

This article's purpose is to guide you through creating a basic security system. I will demonstrate how you can protect your web app and manage the users that are permitted to access it. A nice feature is that the login procedure is done using the Maplaz (/), so you wouldn't have to deal much with user management and credentials.

At the end of this article, you should have a secured web app. You could get information on who tried to login to your site, and you could choose who has access to your app and who doesn't. As mentioned, the login procedure would be with the Maplaz (although you could define Username-Password interface).

Recommended way to study

Firstly, read the introduction (you are already in the right direction).

I recommend dividing the System Setup into two parts:

- Local Setup where everything is running on your computer. All of the routes would be localhost, and the setup would use docker-compose.
- OpenShift Setup all of the components would be uploaded to OpenShift (OS). This is a system that is much more production ready, but is harder to set up because of OpenShift networking. In addition to the previous Local Setup, you would have to build Service and Route objects in OS.

This way is recommended because you would understand the system better, and you could handle the issues better (and there will be issues, trust me).

If you need to strengthen your OpenShift skills, you are invited to read my OpenShift Guide (shameless self-advertising #1).

Basic terminology

- Authentication The processs of confirming the identity of the user \ website.
- Authorization The process of granting permissions to a certain user. For example allowing only users in the admin group to enter a site, while blocking it for other users.
- IDP Identity Provider, a database that holds user information in a specific network. It can authenticate the user that tries to log
 in. myIDP is the IDP of the "my" network, we will use it.
- OAuth 2.0 A common authentication protocol in which a certain site gets user information from an IDP, without giving the site your sensitive
 information.
- OIDC OpenID Connect, it's a protocol that expands OAuth 2.0. For our purpose it is the same as OAuth 2.0 (although there is a difference!).

Architecture Overview

Components

There are two main components at the heart of our security system:

Keycloak – An opensource app that provides a security layer. It will manage the login page, OIDC authentication flow with myIDP, authorization, user-management and so on. Basically everything regarding the configuration of your security layer.



If Keycloak approves the user enterence, Nginx will forward you to the requested site.

In addition you would have to connect Keycloak to a PostgreSQL DB, and have an app that you want to protect.

Flow

The flow of the request is sketched in this great diagram that I made (it's beautiful, I know). The parts we would have to set up are the **brown parts** and the arrows, assuming you already have an app \ website.



You can think about the flow this way:

- 1. The Tiger (user) wants to get the Rabbit (protected app) because it is hungry (needs to work). The Tiger has to ask for permission from the wise Fox (Nginx), because the Fox is in charge of all the network communication in the forest.
- 2. The Fox (Nginx) transfers the request to the Lion (Keycloak).
- 3. The Lion (Keycloak) doesn't recognize the Tiger (user), so he turns to his wife the Lioness (myIDP). The Lioness (myIDP), being much smarter than her husbend the Lion, recognizes the Tiger (user) and **authenticates** him using OIDC protocol.
- 4. The Lion (Keycloak) can now authorize the Tiger (user). He tells the Fox (Nginx) that all is good.
- 5. The Tiger (user) is now both authenticated and authorized. So the Fox (Nginx) forwards him to the Rabbit (website).
- 6. Bonus: Everyone is happy (except maybe the Rabbit's kids).

BTW the Keycloak DB is not a part of the flow, it's just that Keycloak can't work without a DB to store configuration and user information.

Local installation

We will use docker compose to upload four components:

- App/website The app you want to upload and protect. I will use a simple python-flask server (you can just as easily use nodejs express server).
- Kevcloak
- Keycloak DB
- Nginx

In the following subsections I will explain the configuration of the important parts. I would refer to the docker-compose yaml file below and the GitHub repository (in the local_setup folder).

Note - Client Registration

In order for the IDP to respect your requests and the whole thing to work, you need to register a client to the myIDP (the IDP). It is described in the Client Registration section.



Docker-Compose YAML

```
version: "3.7"
networks:
 mynetwork:
   name: mynetwork
    attachable: true
services:
  postgres:
           image: <registry>/rhscl/postgresql-10-rhel7@sha256:
13703497b40861b4c0563020c17b9bb2d68f6956ed53312577c7ef09e8ff9fa7
     volumes:
      - keycloak_data:/var/lib/pgsql/data
    environment:
      POSTGRESQL_DATABASE: keycloak
     POSTGRESQL_USER: keycloak
     POSTGRESQL_PASSWORD: password
   ports:
      - 5432:5432
   networks:
     - mynetwork
  kevcloak:
    image: <registry>/my-keycloak/keycloak-certs@sha256:
3a13fd0e01587d2790f4695b5129b20f1112039742cb7c96831f529dad10a202
    environment:
      DB_VENDOR: POSTGRES
      DB_ADDR: postgres
     DB_DATABASE: keycloak
      DB_USER: keycloak
      DB_SCHEMA: public
      DB_PASSWORD: password
     KEYCLOAK_USER: admin
     KEYCLOAK_PASSWORD: password
      # Uncomment the line below if you want to specify JDBC parameters. The parameter below is just an
example, and it shouldn't be used in production without knowledge. It is highly recommended that you read
the PostgreSQL JDBC driver documentation in order to use it.
      #JDBC_PARAMS: "ssl=true"
    ports:
      - 3333:8080
    depends_on:
      - postgres
   networks:
      - mynetwork
  nginx:
   image: <registry>/my-keycloak/ngnix_base@sha256:
5fb160a510eeb11b72a2e17de0d6b8da3c05c4d51e6f60afca10733b6b655d52
  logging:
     driver: "json-file"
      options:
       max-size: "10m"
    ports:
      - "90:90"
    volumes:
      - C:\Users\<your_user>\Desktop\SSO\local_setup\nginx-default.conf:/etc/nginx/conf.d/default.conf:ro
    environment:
     KEYCLOAK_URL: '"http://host.docker.internal:3333/auth/realms/myrealm/.well-known/openid-configuration"'
      KEYCLOAK_LOGOUT: '"http://host.docker.internal:3333/auth/realms/myrealm/protocol/openid-connect/logout?
redirect_uri=http://localhost/"'
      LOCATION1: /whatever
      LOCATION2: /
      PROXY_URL: http://web:5000/
```



W W W.UPDF.COMNT_ID: '"nginx"' CLIENT_SECRET: '"efbb35c8-6984-460a-b55f-7f1f025acb2f"' networks: - mynetwork

Note - Docker Volume on Windows

In Windows, before creating a volume in docker, you need to configure it in Docker-Desktop. In Dcker-Desktop go to Settings ResourcesFile Sharing, there add your desired volume directory.

Keycloak

We will use an image of Keycloak that I modified, it is the *keycloak-certs* repo. The important modification is the installation of the myNET certs, so that Keycloak can communicate in HTTPS (basically it adds the myNET CA's to Keycloak trusted CA's).

Create a Realm and a Client

After you ran the Keycloak container, open it in the web browser (with the current docker-compose it's in *http://localhost:3333/*). Click on *Admin-Console* and enter your admin credentials. Then you should see the following screen:

					🛓 Admin 👻
Master ~	Master 👕				
	General Login	Keys Email Themes Cach	Tokens Client Registration	Security Defenses	
🚻 Realm Settings	* Name	master			
Clients	Display name	Kevrloak			
🚲 Client Scopes	bispidy name	Reyclook			
📰 Roles	HTML Display name	<div class="kc-logo-text">Keycloa</div>	k		
🚞 Identity Providers	Frontend URL 😡				
User Federation	Enabled @	ON			
Authentication					
	User-Managed Access	OFF			
Groups	Endpoints @	OpenID Endpoint Configuration			
Licore		SAML 2.0 Identity Provider Metadata		_	
 Osers Cossions 		Saus Cancel			
Sessions		Jave Cancer			
Evens					
Import Eveent					
L's Export					



Click on Macter in the upleft corner and choose "Add realm", give it some name. Then click on "Clients" in the sidebar and choose "create" in order to create a new client, give it an ID (I like to call it "nginx"). The client defines inner-configuration inside a realm, you should have a unique client for each application you protect.

In the client configuration page, configure it to look like this:

		1 Admin Y
Myrealm 🗸	Clients → ngirx	
	Nginx 🝵	
W Realm Settings	Settings Roles Client Scopes O Mappers O Scope O Revocation Sessions O Offline Access O Installation O	
😚 Clients	Client ID @ nrinx	
🛞 Client Scopes		
📰 Roles	Name 🛛	
Identity Providers	Description @	
User Federation	Enabled ON	
Authentication	Consent Required OFF OFF	
	Login Theme 😡 🔻	
🐁 Groups		
🛓 Users		
 Sessions 	Access Type Confidential	
Events	Standard Flow Enabled ON	
	Implicit Flow Enabled OFF	
L's export		
	Enabled @	
	Service Accounts ON Enabled ()	
	Authorization Enabled	
	Root URL ®	
	Valid Redirect URIs Inttp://localhost4	
	Base URL @	
	Admin URL Ø	
	Wah Origins @	
	> Fine Grain OpenID Connect Configuration @	
	> OpenID Connect Compatibility Modes @	
	> Advanced Settings @	
	> Authentication Flow Overrides 🚱	
	Save Cancel	

There are loads of configuration options for the client, the important ones for us are:

- Access Type Choose public if you are testing. Otherwise choose confidential and save the Client Secret from the credentials tab (it appears after you hit save).
- Valid Redirect URI's In these you specify the valid paths that the client receives. In the local version it should be http://localhost/* and http://localhost/* and http://localhost/* and http://localhost/* and http://localhost/ and http://
- Under Authentication Flow Overrides there are the default client Authentication Flow's. We will reconfigure it in the Configuring Authorization s
 ection.

Tip - Client Configuration

Under Roles you can create client specific roles. If you make access type confidential and enable Authorization than a new sub-tab opens. It is the Authorization tab. with it you can control your client authorization much more flexibly (for example block a certain role from a specific route). It requires extending Keycloak with scripts (to enable a policy-enforcer) and diving into the Keycloak world.

Create and configure the IDP

The identity provider (IDP) is the part of the system that will allow users to log in without entering a username and password, it will get their user info using OIDC and the Maplaz. Frankly it's the coolest part of this system. The steps to configure

- 1. In the sidebar choose "Identity Providers" and create a new one using the dropdown. Choose "OpenID Connect v1".
- In the IDP configuration, first you should choose an Alias. Notice the Alias is not an unimportant name! The unimportant name is the Display Name, the Alias goes automatically into the Redirect URI (above). That is very important, because we need to configure the OIDC client to accept our particular Redirect URI.
- 3. Enter the following address that are needed for the OIDC protocol taken from this json (https://myIDP/auth/realms/prod/.well-known

/openid-configuration). In case you use Mamram and not Marganit, change "mr" to "mm".

"Authorization URL": "https://myIDP/auth/realms/prod/protocol/openid-connect/auth "Token URL": "https://myIDP/auth/realms/prod/protocol/openid-connect/token" "User Info URL": "https://myIDP/auth/realms/prod/protocol/openid-connect/userinfo"



Authentication" choose "Client secret sent as post".

5. Enter your Client ID and Client Secret

Finally, your IDP should look like this:

Important Notes

- You need to make sure the client-id and the redirect-uri are known to the IDP. Registration to the IDP is described in the Client Registration section.
- .
- In order to actually **block users**, I refer you to the last section in this article called Configuring Authorization. After logging in with the IDP it would redirect you to the *redirect URI* you gave it with HTTPS. For the local installation, I reccommend to just remove the 's' from 'https' when you get redirected.

Nginx

In the local setup the Nginx image I used is the nginx_base image in the our repo my-keycloak. It is just Nginx version openresty, with Lua modules

that support OIDC. You can find a reference for the Dockerfile in git.

You need to configure the conf file using the env variables. In the local version we are going to add to the main conf file (that is in the The conf file is in /etc/nginx/conf.d/default.conf, you can find the conf file also in the GiHub repository under local_setup. The file is this one:



Tip - Nginx conf

Actually this is not the main Nginx conf file. This is just added to the main conf, in our *openresty* version of Nginx, the main conf file is: /usr/local /openresty/nginx/conf/nginx.conf .

```
server {
  listen
             90 default_server;
  root
           /opt/nginx/html
  resolver 127.0.0.11 valid=1s ipv6=off
  access_by_lua
   local opts =
     redirect_uri_path = "/",
     accept_none_alg = true,
discovery = "http://host.docker.internal:3333/auth/realms/<myrealm>/.well-known/openid-configuration",
     client_id = "<client_name>",
     client_secret = "<secret from keycloak nginx client>",
     redirect_uri_scheme = "http",
     logout_path = "/logout",
     redirect_after_logout_uri = "http://host.docker.internal:3333/auth/realms/<myrealm>/protocol/openid-
connect/logout?redirect_uri=http://localhost/",
     redirect_after_logout_with_id_token_hint = false,
     session_contents = {id_token=true}
   -- call introspect for OAuth 2.0 Bearer Access Token validation
   local res, err = require("resty.openidc").authenticate(opts)
   if err then
     ngx.status = 403
     ngx.say(err)
     ngx.exit(ngx.HTTP_FORBIDDEN)
   end
  # I disabled caching so the browser won't cache the site.
                    0
  expires
  add_header
                    Cache-Control private;
  location / {
  location /web {
   proxy_pass http://web:5000/;
  # redirect server error pages to the static page /40x.html
  error_page 404 /404.html;
    location = /40x.html {
  }
  # redirect server error pages to the static page /50x.html
  error_page 500 502 503 504 /50x.html
    location = /50x.html {
  }
}
```

Notice:

- The discovery url is set to the Keycloak container under a realm <myrealm>.
- Client name should be specified, (I usually use 'nginx').
- If your client is defiend as confidential than you should also provide the client secret.
- I used port 90 becasue 80 was taken.
- In the current setup, because the proxy is defined under location /web, you need to go to http://localhost:90/web in order to be redirected.

Result

If all is configured correctly, when you enter to the Nginx url (mine is http://localhost:90/web), you should be redirected to a Keycloak login page that looks like this:



0																								>
÷	G	仚	Note	secure	hos	.docke	er.inte	rnal:33														6	Incognito	
¥		3	>	Pro	d 📘			🕙 Exte	end	🧿 L														
											Ν.Λ.	vD	c /	A I I	N /									
											IVI	IR	C <i>1</i>	≺L	IVI									
			Г																					
													σI	n										
												LU	6'	÷.,										
				User	name	or emai	il					1					IDP	display	name					
				Ľ							 	J												
				Pass	word																			
								Log I	In															

Then you click on the IDP in the right, and it should redirect you. If you don't use https, you need to change *https http* in the redirected url (as noted in the important notes). Then, you should be redirected to your app.

If you want to actually block users, look at the Configuring Authorization section.

OpenShift installation

After dealing with the Local Installation, you should have basic familiarity with the system (honestly i'll be impressed if you read and followed this longass article).

The OpenShift installation shouldn't be much harder if you managed all of the previous stuff, it would mainly require re-configuration, and knowladge of OS (OpenShift) networking with Services and Routes. If your'e OS is rusty, you can read my OpenShift Guide (shameless self-advertising #2).

Upload to OpenShift using a Template

I am going to use an OpenShift Template file in order to create all of the objects needed to make the system work (4 Deployments, 4 Services and 2 Routes).

Tip - Using OC and Templates

You can upload the template file using the following OC (OpenShift client) command:

oc process -f path/to/template.yaml | oc create -f -

And since the objects created in the Template get their labels from the template, you can delete them all using:

oc delete all -l <label_name>=<label_value>

The Template file I used:

OpenShift Template

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
    name: test-keycloak
# These labels would pass to all objects created in this template!
labels:
    group: keycloak
objects:
    - kind: Deployment
    apiVersion: apps/v1
    metadata:
        name: ${KEYCLOAK_NAME}
```



name. \${KEYCLOAK_NAME}

```
spec:
   replicas: 1
   selector:
     matchLabels:
       name: ${KEYCLOAK_NAME}
    template:
     metadata:
       labels:
         name: ${KEYCLOAK_NAME}
      spec:
       containers:
          - name: ${KEYCLOAK_NAME}
           image: ${KEYCLOAK_IMAGE}
           ports:
             - containerPort: 8080
               protocol: TCP
            env:
              - name: DB_VENDOR
               value: POSTGRES
              - name: DB_ADDR
               value: ${DB_SER_NAME}
              - name: DB_DATABASE
               value: keycloak
              - name: DB_USER
                value: ${DB_USERNAME}
              - name: DB_SCHEMA
               value: public
              - name: DB_PASSWORD
                value: ${DB_PASSWORD}
              - name: KEYCLOAK_USER
               value: ${KEYCLOAK_USERNAME}
              - name: KEYCLOAK_PASSWORD
                value: ${KEYCLOAK_PASSWORD}
              - name: PROXY_ADDRESS_FORWARDING
                value: 'true'
- kind: Service
 apiVersion: v1
 metadata:
   name: ${KEYCLOAK_SER_NAME}
 spec:
   ports:
     - port: 8080
       targerPort: 8080
       protocol: TCP
    selector:
     name: ${KEYCLOAK_NAME}
- kind: Route
 apiVersion: route.openshift.io/vl
 metadata:
   name: ${KEYCLOAK_SER_NAME}-route
 spec:
   host: ${KEYCLOAK_URL}
```



```
кına: service
     name: ${KEYCLOAK_SER_NAME}
   tls:
     termination: edge
- kind: Deployment
 apiVersion: apps/v1
 metadata:
   name: ${DB_NAME}
   labels:
     name: ${DB_NAME}
 spec:
   replicas: 1
   selector:
     matchLabels:
       name: ${DB_NAME}
   template:
     metadata:
       labels:
         name: ${DB_NAME}
      spec:
       containers:
          - name: ${DB_NAME}
           image: ${DB_IMAGE}
            env:
              - name: POSTGRESQL_DATABASE
               value: keycloak
              - name: POSTGRESQL_USER
               value: ${DB_USERNAME}
              - name: POSTGRESQL_PASSWORD
                value: ${DB_PASSWORD}
           ports:
              - containerPort: 5432
               protocol: TCP
            volumeMounts:
              - name: "keycloak-data"
               mountPath: /var/lib/pgsql/data
       volumes:
         - name: "keycloak-data"
           presistentVolumeClaim:
             claimName: "keycloak-data"
- kind: Service
 apiVersion: v1
 metadata:
   name: ${DB_SER_NAME}
 spec:
   ports:
     - port: 5432
       targerPort: 5432
       protocol: TCP
    selector:
     name: ${DB_NAME}
```

 $\ensuremath{\tt \#}$ There is a need to check whats going on with the volume and the persistent volume claim (at least in prod)!

```
WWW.UPDF.COM
         KING: Deployment
         apiVersion: apps/v1
         metadata:
          name: ${NGINX_NAME}
           labels:
            name: ${NGINX_NAME}
         spec:
          replicas: 1
           selector:
            matchLabels:
              name: ${NGINX_NAME}
           template:
             metadata:
              labels:
                name: ${NGINX_NAME}
             spec:
               containers:
                 - name: ${NGINX_NAME}
                  image: ${NGINX_IMAGE}
                   env:
                     # NOTICE: The current dockerfile overrides the system env vars. Stuff like $host wont work.
                     - name: KEYCLOAK_URL
                      value: '"https://${KEYCLOAK_URL}/auth/realms/myrealm/.well-known/openid-configuration"'
                     - name: KEYCLOAK_LOGOUT
                       value: '"https://${KEYCLOAK_URL}/auth/realms/myrealm/protocol/openid-connect/logout?
     redirect_uri=http://localhost/"'
                     - name: PROXY URL
                       value: http://${APP_SER_NAME}
                     - name: LOCATION1
                       value: /web
                     - name: LOCATION2
                      value: /
                     - name: CLIENT_ID
                      value: '"nginx"
                     - name: CLIENT_SECRET
                       value: '"49165460-e763-4ad5-a1f3-184504f7dbe3"'
                     - name: DNS
                       value: resolver 172.30.0.10 ipv6=off;
                   ports:
                     - containerPort: 80
                      protocol: TCP
       - kind: Service
         apiVersion: v1
         metadata:
          name: ${NGINX_SER_NAME}
         spec:
          ports:
             - port: 80
              targetPort: 8080
              protocol: TCP
           selector:
             name: ${NGINX_NAME}
```



```
apiversion. route.openshift.io/vl
    metadata:
     name: ${NGINX_SER_NAME}-route
    spec:
     host: ${APP_URL}
      to:
       kind: Service
       name: ${NGINX_SER_NAME}
      tls:
        termination: edge
  - kind: Deployment
   apiVersion: apps/v1
    metadata:
     name: ${APP_NAME}
      labels:
       name: ${APP_NAME}
    spec:
     replicas: 1
      selector:
       matchLabels:
         name: ${APP_NAME}
      template:
        metadata:
         labels:
           name: ${APP_NAME}
        spec:
          containers:
            - name: ${APP_NAME}
             image: ${APP_IMAGE}
             ports:
               - containerPort: 5000
                 protocol: TCP
  - kind: Service
   apiVersion: v1
    metadata:
     name: ${APP_SER_NAME}
    spec:
      ports:
        - port: 80
         targetPort: 5000
         protocol: TCP
      selector:
       name: ${APP_NAME}
parameters:
  - name: DB_NAME
   value: keycloak-db
  - name: KEYCLOAK_NAME
   value: keycloak
  - name: NGINX_NAME
   value: keycloak-ngnix
  - name: APP_NAME
   value: keycloak-testapp
```

```
- name: KEYCLOAK_IMAGE
value: <registry>/my-keycloak/keycloak-certs@sha256:
```



WW.UPDFbGOMb4aa5e6db4588a5247f3f1e948cdf2315ae0672a433ac8c3bc889f

<pre>- name: DB_IMAGE value: <registry>/rhscl/postgresql-10-rhel7@sha256: 13703497b40861b4c0563020c17b9bb2d68f6956ed53312577c7ef09e8ff9fa7</registry></pre>
<pre>- name: NGINX_IMAGE value: <registry>/my-keycloak/configured_ngnix@sha256: 24a4a8bb3f139009620c80c4482662aedad85408f1c4bc0c36fddbe83cd65e9c</registry></pre>
- name: APP_IMAGE value: <registry>/my-keycloak/test-app@sha256: 54a022bfe2c4bb2599ed6cbf39f7c3f1f42292b8df79fb75d953ff766f54e21b</registry>
- name: DB_SER_NAME value: keycloak-db-service
- name: KEYCLOAK_SER_NAME value: keycloak-service
- name: NGINX_SER_NAME value: keycloak-nginx-service
- name: APP_SER_NAME value: keycloak-app-service
- name: BB_03ERRAME value: admin - name: KEYCLOAK USERNAME
value: nadavDaKing - name: DB_PASSWORD
value: admin - name: KEYCLOAK_PASSWORD
value: 1tob - name: APP_URL value: keycloak-test-app.ocp4 myNFT
<pre>- name: KEYCLOAK_URL value: keycloak-admin.ocp4.myNET</pre>

DONT PANIC from the Template's size, it's just the configuration for 4 Deployments, 4 matching Services and 2 Routes. If you'll break it to its parts you'll find they're quite simple.

Notice:

- Nginx conf file is now built from environment variables, and it is the real conf file and not the fake one (as mentioned in the Local Setup Nginx section).
- Nginx discovery is the Route of Keycloak, instead of the Keycloak Service (I couldn't manage to make it work with the service, it had resolverdns issues. If you manage to make it work pls tell me).
- Nginx proxy is the service of the Protected App.
- Keycloak's database address is changed to the database service.
- The ports had changed because all of the Services and Routes. Now you just enter the routes at regular port 80.

If all worked well, you should be able to get the Admin Console from the keycloak-service-route. And when you go to the keycloak-nginx-service-route URL you should see the following error:

					🕻 Implementing 🗙 🛛 🕋 IDFCTS ID 🛛 🗙 🛛 🙆 Deployments 🗙 🧑 https://keyclo 🗙 🌀 Welcome to K 🗙 🛛 🦊 openshift_sets 🗙 🕇		٥	×
÷		С	仚		keycloak-test-app.apps.ocp4.np.mr.idf.cts/	* 🔅	Θ	
+	×	¢	3		🚬 📙 Prod 📕 NonProd 🔇 Extend 🐠 Log in to myrealm			
coul	d not	: dec	ode J	SON 1	from Discovery data: response indicates failure, status=404, body={"error":"Realm does not exist"}			

Which means that you have not created a Keycloak realm yet.

If you got so far, honestly, either you are crazy or your commander is crazy. Either way you deserve a prize.

Extra Configuration



we drawe to recoming are some of the realm, client, IDP and IDP-client settings.

Open up Keycloak admin-console (using the previously defined route) and enter the admin credentials ("nadavDaKing", "1to6"). Create a realm and and a client similarly to the Local Setup configuration. Now, in the **Valid redirect URI's**, enter *http://keycloak** and *https://keycloak.*

For the IDP configuration, again follow the steps in the Local Setup IDP configuration. Make sure to copy the new Redirect URI and to put it in the

valid IDP-Client URI list, as explained in the myIDP Client Registration.

If all is done correctly, you should be able to enter the Nginx route, get a Keycloak login prompt, login with the IDP and then be redirected to your Protected App!

Now - if you haven't done it already - the last step is to actually block the users using authentication flows. It is explained in the Configuring Authorization section.

Authenticate and Authorize

myIDP Client Registration

As mentioned, Keycloak communicates with the myIDP (our IDP) using OIDC protocol. The protocol requires that you define a client, and send a

valid redirect URI (so that the IDP would know where to redirect you after it authenticates you).

So in order for the whole thing to work, you need to register a client at your IDP. Fortunately for us, the great team of myIDP made it easy for us (tnx guys!).

Enter their portal at https://portal.myNET/ and create a new client. You should be redirected to the new client page:

TEXT



- Client ID Choose a name to your likings, you'd enter it at the IDP configuration.
- Client Secret Choose a secret (a password), you'd enter it at the IDP configuration. BTW this is a great place for easter eggs, I reccommend using base64.
- Redirect URI's This is the place to enter the redirect URI's you saw at the Keycloak IDP configuration page. Notice that you can insert
 multiple addresses. Also notice that the Local Setup and the OpenShift Setup require different redirect URI's. My configuration looks like this:

Click submit and your'e done!

Configuring Authorization (blocking users)

After reading this enormous article, trying to set up the system, failing, crying, looking up for other occupations, and finally somehow making this work – you might wonder how to actually block users from entering your site (or is it just me?). This is the authorization process, and Keycloak has a lot of ways to accomplish that. I will show a simple way I figured it out, but if you'd like to implement a more complex authorization, I recommend reading about Keycloak's roles, polices, scopes, groups and policy enforcers.

The way we are going to implement authorization is using Keycloak Authentication Flows. These are the processes every user is going through when he logs in (without him/her knowing). We are going to assign every new user that logs in an *Enemy* role, so that he can't get into the site. Than, the Admin (King) can view the user and decide if he is really an *Enemy* of the realm or not. If the Admin (King) decides to let the new traveler pass, he simply needs to unassign the user from the *Enemy* role.

First, let's create a Role: on Keycloak admin-console, click on *Roles* in the left sidebar. Then *Add Role*, name him something, I'm going with *Enemy*. Then go to the *Default Roles* tab and make sure he is the only *Realm Defauly Roles*:

	KEYC LOAK				
Myre	alm 🗸	Roles			
Config		Realm Roles Defau	ult Roles		
111 12 12 12 13 12 13 12 13 12 13 14 14 14 14 14 14 14 14 14 14 14 14 14	Realm Settings Clients Client Scopes Roles Identity Providers User Federation Authentication	Realm Roles Client Roles	Available Roles © King offline_access Pessant urma_auchorization Add selected > Select a client	Realm Default Roles @ Enemy « Remove selected	×
Mana					
24	Groups				
2	Users				
0	Sessions				
Ê	Events				
ß	Import				
	Export				

Follow these steps to configure the Authentication Flows that we will use for authorization:

- 1. Select the Authentication tab from the left sidebar. You should see a browser Authentication Flow schema. That means that every regular browser log in attempt would go through these processes.
- 2. Click on Copy, in order to create a copy of this Flow and name him whatever you'd like (I chose Browser w Roles).
- 3. Click on Add flow in order to add a new Authentication Sub-Flow and give it some name. On the new subflow that you created, make sure con ditional is clicked.
- 4. Click on the Actions dropdown of the new sub-flow, choose Add execution. Then choose Condition User Role from the drop down.



W. UP DF GRMon REQUIRED in the new condition, then go to Actions and hit config.

- d from logging in. So write down *Enemy*, or the lame role name you chose earlier. Hit save.
- 7. Repeat steps 2-6 for the First Broker Login Flow. Also create a new Flow (using new) and apply steps 3-6 to it.

At the end of this process, you should have created 3 new Flows, two copies and one new. They should look like these:

to man here	1										
/yrealm ~	Authentica	ition	Actions Pr	word Balia : CTT	Dolloy Water at a		h Author 5		Dollo - C		
	Hows Bind	iings kequired	Actions Passv	vord Policy OTP	Policy WebAuthn P	olicy 🥹 🛛 We	spautnn Pa	sswordless i	Policy 🥹		
🚻 Realm Settings	Browser W Role	s v O					New	Copy Dele	te Add	execution	dd flow
Clients	Auth Type					Requiremen	nt 🛞				Actions
Client Scopes						REQUIRED	ALTERNAT	IVE DISABI	LED		~
Identity Providers	Kerbero	IS				REQUIRED	ALTERNAT		LED		Actions ~
User Federation	∧ ∨ Identity	Provider				0	۲	0			Actions
Authentication	Redirector					REQUIRED	ALTERNAT	IVE DISABI	LED		×
	Forms Browser	r W Roles				REQUIRED	ALTERNAT	IVE DISABI	LED CO	ONDITIONAL	Actions
			Username Passy	word Form		۲					Actions
· Urour			Browner W Pole	Recourses		REQUIRED	0	0			Actions
 Sessions 		Condi	tional OTP			REQUIRED	ALTERNAT	IVE DISABI	LED CO	ONDITIONAL	~
Events					Condition - User	REQUIRED	DISABLE	ED			Actions
⊡ Import					UTP Form		0	0			Actions
Export						REQUIRED	ALTERNAT	IVE DISABI	LED		×
	Roles					REQUIRED	ALTERNAT	IVE DISABI	LED CO	ONDITIONAL	Actions
			Condition - User	Role (Enemy		۲	O DISABLE	ED			Actions
		denie	d)			REQUIRED					×.
KEYCLOAK											Nadavdaking
RETELOAR											
yrealm ~	Authenticatio	on									
nfigure	Flows Bindings	s Required Actions	Password Policy	OTP Policy We	Authn Policy @ WebAut	hn Passwordless	Policy 😡				
Realm Settings	Access By Role	¥ 0						New Cop	Delete	Add execution	Add flow
Olients Olient Scones	Auth Type Roles3				Requirement	TERNATIVE			ONDITIONA	u Arr	inns -
Roles		🔨 🗸 Conditio	n - User Role (Enemy de	anied)	REQUIRED O DI	ABLED	- Unimpeer		ononio	Act	ions 🗸
inage ≩L Groups ≜Users ⊙ Sessions Events											
Anger 4⊾ Groups ≜ Usersions ∰ Events ∭ Import C, Export											
Al Groups 4. Groups 5. Seasons ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰	Authenticatio	in								Ŧ	Nadavdaking
lager ≩ Ginups ≧ Users Sassions ∭ Innorn © Export ■ KEYCLOAK realm v 1007	Authenticatic	۲۲ Required Actions	Password Policy	OTP Policy Web	Authn Policy @ WebAut	nn Passwordless	Policy @			*	Nedevdeking
logar ≩ Ginups ≧ Sastons © Sastons ∰ Evens © Export ■KEYCLOAK realm v ter ter tot	Authenticatio	011 Required Actions Roles V 0	Password Policy	OTP Policy Web	Auth Policy@ WebAut	nn Passwordless	Policy @	New Coo	ry Delete	1 Add execution	Nadavdaling Add flow
Auger 4 Groups 5 Users 5 Sessions 6 Events 2 Export IKEYCLOAK realm 3 Trealm 4 500 100 100 100 100 100 100 100	Authenticatio Fow Bindings Free Broker Legin W	D Required Actions Roles • •	Password Policy	OTP Policy Wet	Auchn Policy @ WebAuc	nn Passwordiess	Policy @	New Cop	y Delete	Add execution	Add from
A Groups 4 Groups 5 Groups 5 Groups 5 Groups 1 Grou	Authenticatio Fows Bindings Frost Broker Lagin Auth Type Profile (Prefer	DD Required Actions Roles V 0	Pessword Policy	OTP Policy Wet	Auth Policy @ WebAut	nn Pesswordless	Policy @ Requirement @ REQUIRED	New Cop	y Delete DISABLEC	Add execution	Add flow Action
A Groups 2. Uohon 5. Sensor 5.	Authenticatio	DD Required Actions Roles v 0	Password Policy	OTP Policy Web	Auchn Policy @ WebAuc	hn Passwordless	Policy @ Requirement REQUIED ®	New Cop ALTERNATIVE	y Delete	Add execution	Add flow Action
All Groups 2 Ubors 3 Integrations 3 Integrations 4 EVECLOAK EVECLOAK EVECLOAK EVECLOAK 2 Integrations 4 Integrations	Authentication	IN Required Actions Required Actions	Password Policy	OTP Policy Wet	Auchn Policy @ WebAuc	hn Passwordless	Policy @ Requirement ® REQUIRED ® REQUIRED	New Cop Alternative	y Delete DISABLET DISABLET	Add execution	Add flow Action Action AL
A Company A Company S Stations S Stations S Stations MEYEVELOAK Treation MEYEVELOAK Market Ma	Authenticatio Flows Bindings Flows The State Light W The The State Light W and The Contemport profile contemport The State Light W Block Linker Castelin O' Linker)N Required Actions Roles v 0	Password Policy	OTP Policy Wet	Austin Policy Ø ViebAus	nn Pesswordless	Policy @ Requirement ® REQUIRED ® REQUIRED	New Cop ALTERNATIVE ALTERNATIVE *	y Delete DISABLED DISABLED	Add execution	Add flow Add flow
A forger 4. Uoors 5. Solors 5. Solors 5. Boors 10. Forger Tealm 10. Solors 10. Sol	Authenticatio Flows Bindings Fred Broker Lagin Auth Type In Stream Profile Stream In Fred Broker Lagin William User Creation Or Living	IN Required Actions Roles V 0	Password Policy	GTP Policy Wet	Austrn Policy Ø WebAust	nn Pesswordless	Policy @ Requirement * REQUIRED & REQUIRED @ REQUIRED	New Cop ALTERNATIVE ALTERNATIVE * ALTERNATIVE	y Delete DISABLET DISABLET DISABLET	Add execution Add execution CONDITION CO	Add flow Action Action Action
All Groups 2. Uson 3. Uson 5. Georet 5. Georet EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIOAK EXECUCIÓA SECUCIÓN EXECUCIÓA SECUCIÓN EXECUCIÓN SECUCIÓN SECUCIÓN EXECUCIÓN SECUCIÓN SECUCIÓN EXECUCIÓN SECUCIÓN SECUCICIÓN SECUCI	Authenticatio	IN Required Actions Rakes Required Actions Rakes	Password Policy	OTP Policy Wet	Auth Policy @ WebAut	nn Passwordless	Policy @ Requirement * REQUIRED * REQUIRED @ REQUIRED	New Cop atternative atternative atternative	y Delete DISABLET DISABLET DISABLET	Add execution Add execution CONDITION CO	Add flow Action Action Action
Ale Groups A Groups A Groups Genome Genome Genome Genome Control MeteryCLOAK MeteryCLOAK Control Genome Geno	Authenticatio	Required Actions Relax	Password Policy	oth Policy Wet	Auchn Policy @ WebAuc	hn Passwordless	Policy @ Requirement * REQUIRED * REQUIRED * REQUIRED * REQUIRED	New Cop atternative atternative atternative atternative atternative	y Delete DISABLET DISABLET DISABLET DISABLET DISABLET	Add execution CONDITION: CONDITION: CONDITION:	Add flow Action Action Action Action Action Action
spr 4 Urons 5 Genome 5 Genome 5 Genome 5 Genome 1	Authenticatio Flow Bindings Frist Braker Lagie With Type In Provide Spreak profile Options profile Options Pro	Rate Regulard Actions Regulard Action Regulard Actions Regulard Actions Regulard Actions Regula	Password Policy	OTP Policy West	Auch Polcy Ø WebAuc	hn Passwordiess	Policy @ Requirement # REQUEED # REQUEED REQUEED REQUEED	New Cop ALTERNATIVE ALTERNATIVE ALTERNATIVE	y Delete Orsablet Disablet Disablet Disablet	Add execution CONDITION CONDITION CONDITION	Add flow Action Action Action Action Action
A Groups General	Authenticatio	D Required Actions Required Actions Reades ♥ Ø	Password Policy	OTP Policy Wet	Autor Policy Ø ViebAut	nn Passwordless	Policy @ Requirement # REQUISED @ REQUISED @ REQUISED # REQUISED	New Cop Altismative	y Delete OISABLEC OISABLEC DISABLEC DISABLEC OISABLEC	Add execution Constraints Con	Add flow Add flow Action Action Action Action
Appr 4 Groups 2 Good 5 Groups 5 Good EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK EXECULOAK E	Authenticatio	Required Actions Reals	Password Policy	CTIP Policy Wet	Austrn Policy @ WebAust	nn Passwordless	Policy @ Requirement * RAQUARD 0 RAQU	New Cop ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE	y Delete DISABLED DISABLED DISABLED DISABLED DISABLED DISABLED DISABLED DISABLED DISABLED DISABLED DISABLED DISABLED	Add execution Controllow Co	Add flow Add flow Action Action Action Action Action Action Action Action Action
lapor la foraça la f	Authenticatio	Required Actions Rekar	Pessword Policy	atti Policy Wet	Austin Policy @ Weshue	nn Passwordless	Policy @ Requirement * RQUIRED NEQUIRED 0 REQUIRED * RQUIRED * RQUIRED *	New Cop ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE	y Delete DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE	Addressonion costorion costorion costorion costorion costorion	Actional States
Appr 4 Deps 5 Sectors 5 Sectors 6 Sectors 6 Sectors 7 Sector	Authenticatio	Resulted Actions Resulted Actions Result v 0 Crarte User 7 Crarte User 7 Crarte User 7 Crarte User 7 Crarte User 7 Crarte User 7 Crarte User 7 Crarte User 7 Crarte User 7 Crarte	Password Policy	OTP Policy Wet	Auch Polcy@ WebAuc	hn Passwordless	Policy @ Requirement # REQUIRED # REDUIRED #	New Cop ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE	y Delete DISABLET DISABLET DISABLET DISABLET DISABLET DISABLET DISABLET	Add rescular to the control of	Add flow Add flow Action
Inger Seriors Serio	Authenticatio	Required Actions Reads P P P P P P P P P P P P P P P P P P P	Password Policy	OTP Palicy Wet	Austri Policy Ø Vieblus	No Resources	Policy Ø Requirement ROURED REQUIRED RED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED REQUIRED RED RED REQUIRED RED RED RED RED RED RED RED RED RED	New Copy ALTERNATIVE * ALTERNATIVE * ALTERNATIVE * ALTERNATIVE *	y Delete DISABLET DISABL	Add execution Add execution Common Com	Add film Add film Add film Add film Add Add Add Add Add Add Add Add Add Ad
All Grages 2. Using 3. Grages 3. Graves 3. Graves 4. Using 4. Const 4. Const 4. Const 4. Const 4. Const 4. Const 4. Const 4. Const 4. Const 5. Const 5	Authenticatio	In Required Actions Rates V 0	Password Policy	OTP Policy Wet	Autor Policy Ø ViebAut	N Resources	Policy @ 	New Copp Attractionative * Attractionative	y Deletes DISABLEC DISABLEC DISABLEC DISABLEC DISABLEC DISABLEC DISABLEC DISABLEC	And execution And execution Common	Add flow Add flow Add flow Addo Addo Addo Addo Addo Addo Addo Ad
A Grages 4 2 Using 3 Inport 5 Service 6 Service 8 Concel 8 Concel 8 Concel 8 Concel 9 Concel 10 Concel 10 Concel	Authenticatio	Required Actions Reals Reals P C C C C C C C C C C C C	Pessword Policy	OTP Policy Wet	Auth Policy @ Weskus	In haswordes	Policy @ Reputerment Regulations Regulati	New Cop ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE	y Deletes DISABLET DISABLET DISABLET DISABLET DISABLET DISABLET DISABLET DISABLET	Add execution Common	Add flow Add flow Addo Addo Addo Addo Addo Addo Addo Ad
All Grays 2 Uotos 3 Uotos 5 Sector 6 Sector 7 Arcan 7 Arcan 7 Arcan 7 Arcan 8 Arcan 8 Arcan 8 Arcan 8 Arcan 9 Arcan	Authenticatio	Required Actions Relate V 0 Create Unique user configuration Health Dentity	Password Policy	UTP Policy Wet	Auctin Policy WebAuc	In Resources	Policy 0	New Cop ALTENATIVE ALTENATIVE ALTENATIVE ALTENATIVE ALTENATIVE ALTENATIVE ALTENATIVE ALTENATIVE ALTENATIVE ALTENATIVE	y Delete DISABLES DISABLES DISABLES DISABLES DISABLES DISABLES DISABLES DISABLES DISABLES	Add execution Add execution Add execution Common. Common. Common. Common. Common. Common.	Add flow Add flow Add flow Action
All Grages 2 Uotes 3 Inspect 3 Inspect 4 Context 4	Authenticatio	Required Actions Required Actions Retex	Password Policy	OTP Policy Wet	Auch Polcy WebAuc WebAuc	In Passourdess	Policy @ Reputement Reputeme	New Copy ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE	P Delete OrisAblet	Add execution Add execution Component Com	Add flow Add flow Addor
All Grange 2. Uotros Secondaria Consensation Consensat	Authenticatio	In Required Actions Relate	Password Policy	OTP Policy Wet	Austin Policy Ø WebAus	In Resourcies	Policy @ Papieraent NOURD 0 8 NOURD 0 0 RQURD 0 0 0 0 0 0 0 0 0 0 0 0 0 0	New Cop Attendative Attendati	y Delete DISABLEE ODISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE	Add execution Add execution Commons	Action Action
All Grays 2 Uotos 3 Input 5 Exercis 3 Input 5 Exercis 4 Uotos 10 EVECLOAK 10 EVECLOAK 1	Authenticatio	Required Actions Reads Reads	Pessword Policy	OTP Policy Wet	Auchn Policy @ WebAus	In haswordes	Policy @ Regultrament Regult	New Cop California ALTERNATIVE California ALTERNATIVE California ALTERNATIVE California Califo	V Debete DISABLE DISABLE DISABLE DISABLE DISABLE DISABLE DISABLE	Add rescular Add rescular Common C	Add flow Add flow Addition
All Grays 2 Uoto 3 Uoto 5 Sectors 5 Sectors 1 Sector 1 Secto	Authenticatio	Required Actions Reaks	Pessword Policy	att Policy Wet	Auch Policy WebAuch	In Resources	Policy Ø Regurement #rcyunes #rcy	New Cope ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE O OLSAELED	y Delete DISABLE ODSABLE DISABLE DISABLE DISABLE DISABLE DISABLE DISABLE DISABLE	Add execution Add execution Common Com	Add flow Add flow Add flow Add flow Address
Al Grays 2 Uorg 5 Seniors 1 Uorg 5 Seniors 1 Const 1 Const	Authenticatio	Required Actions Relate V Required Actions Relate Relate Required Actions Relate Required Actions Relate Relate Relate Relate Re	Password Policy	OTP Policy Wet	Auch Polcy Weekut		Policy @ Regularment #RQURED #RQURED #RQURED %RQURED	New Cope ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE ATTENATIVE O ATTENATIVE O O O O O O O O O O O O O	y Delete DISABLE DISABLE DISABLE DISABLE DISABLE DISABLE DISABLE DISABLE	Add execution Add execution Add execution Comprise	Add freework Add freework Addition
All Groups 2. Uohons 3. Groups 4. Uohons 5. Spect Teating 4. Constant 4. Constant 4. Constant 4. Constant 4. Constant 4. Constant 5. Constant 4. Constant 5. Con	Authenticatio	In Required Actions Relate	Password Policy	OTP Paticy Wet	Austin Policy WebAust WebAus	In Passwordess	Policy @ Regurament #EQUESD 0 0 0 0 0 0 0 0 0 0 0 0 0	New Cop SATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY ATEBAATHY	y Dadres Disabile Disabile Disabile Disabile Disabile Disabile Disabile Disabile Disabile Disabile	Add execution Common	Action Action
All Groups 2 Uoho 3 Input 5 Exercs 3 Input 5 Exercs 3 Input 5 Exercs 3 Input 5 Exercs 3 Input 5 Exercs 3 Input 5 Exercs 5	Authenticatio	Required Actions Read	Pessword Policy	OTP Policy Wet	Auch Policy Webus	In haswordes	Policy @ Reguirement, REQUIRED 0 REQUIRE	New Cop ALTERNATIVE ALTERNATI	P Desiste O DisABLIE O	Add rescular Add rescular Common C	Addef flow Addef flow Addef flow Addef flow Addef flow Addef
All Grays 2 Uoto 3 Inport 5 Sectors 1 Sectors 1 Sector 1 Sec	Authenticatio	Required Actions Relax	Password Policy	OTP Policy Wet	Auch Policy WebAuc	In Resourceless	Policy 0	New Cope ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE ALTERNATIVE CO ALTERNATIVE CO SCIENTIFIC ALTERNATIVE CO SCIENTIFIC CO SCIENTIFI	y Deletes DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE DISABLEE	▲ Add rescution ▲ Add rescution ■	Add flag Add flag <t< td=""></t<>

Finally, we'd have to tell the client and the IDP to use our modified Flows instead of the default ones:

- Go to the IDP configuration, in the First Login Flow tab choose the modified first broker flow that you created (first broker flow w roles). In the
 Post Login Flow choose the new Flow you created (Access by Role).
- In the client configuration page, under Authentication Flow Overrides, choose the modified Browser flow that you created (Browser w Roles). NOTICE!! Currently the Browser Flow doesn't work, so don't change it. You can go on without it and nothing would change (the browser flow modifications are only there in order to deny username-password login). When I will remember how to make this work i'll add it.



We are finally done. Go to the nginx-route and get to the Keycloak login page. Click on the IDP provider and wait, you should be redirected to an "invalid login attempt". That means that Keycloak recognized you as an Enemy and denied your access.

Now enter as admin, and go to the Users tab from the sidebar. Click on View all users and you should see your user. Click on it and go to Role Mapping, change the user Assigned Roles and remove Enemy from there. It should look like this:

							💄 Nadavdaking 👻
Myrealm 🗸	Users → s8628617						
Configure	Details Attribute	s Credentials	Role Mappings	Groups Cor	nsents Sessions I	dentity Provider Links	
Realm Settings Clients	Realm Roles	Available Roles @	Assi	gned Roles ©	Effective Roles @		
🚓 Client Scopes 📰 Roles		Enemy offline_access uma_authorization	1				
☐ Identity Providers User Federation		Add selected >	« R	emove selected			
Authentication	Client Roles	Select a client			¥		
Manage							
🛓 Groups							
💄 Users							
 Sessions 							
🛗 Events							
Import							
Export							

Then try to go to nginx-route again. If you can't get the login page, you should probably delete your cookies and cache (if it still doesn't work, go to *Real m Settings Tokens* and change the access token lifespan and other stuff to 1 minute). Try to log in with the IDP provider.

It should work. Probably (:

By:

