


**KeyCloak
(Rocket.Chat)
SSO**

**Dabble in...
Extra Edition2**



**KeyCloak
(Rocket.Chat)
SSO**

**Dabble in...
Extra Edition2**

目次

目次	4
■ はじめに	5
■ 今回のテーマ	5
■ 構築環境	6
■ KeyCloakのインストールと構築	7
■ Rocket.Chatの設定	18
■ おわりに	21
■ あとがき	21

■はじめに

この本は、様々な(主にオープンソースの)ソフトウェアを導入したり実際に使ってみる本です。基本的には興味優先のことが多いですが、なるべく実用に供する参考になるように、あるいは試してみたい方が容易に試用可能なようにする方向で実施します。

どちらかというと「機能的な側面」を優先する記事になるケースが多いです。なるべく補足を入れるつもりですが、現状の記事内容をそのまま実装環境として適用するのは問題がある場合も多いのでご注意ください。

また、当然ながらホスト名、ドメイン名、IPアドレス、アカウント名、などのユニークである必要のある情報はぼかしていたり、参考的なものに変更してあったりしますのでその点はご了承下さい。

また、当方は一応インフラ周辺の技術者ではありますがコード書きではありません。

そのため、いわゆるプログラミング的な側面の内容については誤りがあったり至らない記述をすることもあり得ます。

お気づきの際には是非ご指摘いただけると幸いです。

なお、題名の「dabble in…」ってのはいわゆる「いっちょかみ」的な意味の言葉らしいです。

Extra Editionでは、そこまで突き詰めていなくても現在進行形でやっていることなどを簡単に取りまとめたものを頒布しております。

ガッツリまとめたやつは、別途正式ナンバリングとして発行していきたいです。

■今回のテーマ

今回は、KeyCloakを用いて、既存の環境へのシングルサインオンを実装したいと思います。

シングルサインオンといいつつ、誌面の関係もあり、今回はRocket.Chatへのログイン設定のみを扱います。

ただ、前回同様、動作させることを優先して行いますので、実環境として使用する場合はセキュリティやログ採取面、バックアップなどの環境をきちんと整える必要があります。

■KeyCloakとは

Redhat社が開発しているオープンソースのIDおよびアクセス管理のためのソフトウェアです。まだ新しいソフトウェアですが、活発にリリースされており、現在11.0.3が直近にリリースされています。

今回はSSOにのみ触れますが、IDブローカリングやSNSログインなどといった機能にも対応しており、非常に高機能なソフトウェアです。

■ 構築環境

構築の際の各種環境について簡単に記載します。

■ 使用ディストリビューションについて

KeyCloakの場合はディストリビューションはそれほど関係ありませんが、今回はCentOSを用いています。

```
# cat /etc/redhat-release
```

```
CentOS Linux release 7.8.2003 (Core)
```

■ 仮想マシンについて

- ・ 2CPU
- ・ 8GBメモリ
- ・ 100GB HDD

を割り当てた仮想マシンで行います。

HDDは適当です。試験運用的にはほとんど容量はいらないでしょう。仮想マシンなので可変サイズなので、適当に割り当てています。

■ ハードウェア環境について

前回から引き続き、メモリ1TB/80CPUのIBMマシンを母艦にしています。

(毎回ちょっとした自慢)

■ KeyCloakのインストールと構築

では、ここからKeyCloakのインストールと構築(およびRocket.Chatへの設定)を進めていきます。

本番環境ではバックアップや安定性、容量などの問題もありますので、MySQL等のバックエンドDBを使いますが、前述通り今回は試験的に動かすことを優先し、内蔵のDBで動かします。

■ CentOS7のインストール

CentOS7をインストールします。最小限でインストールしています。詳細は割愛します。

■ アップデート

```
sudo yum update -y
```

お決まりのアップデートをしておきます。

■ JDK8のインストール

keyCloakを使うのに必要なJDK8をインストールします。

```
sudo yum install java-1.8.0-openjdk
sudo yum install wget
```

■ KeyCloak本体のダウンロードと展開

/opt配下にKeyCloakを配置します。

なお、KeyCloak自体は展開した場所以外はDBのみしか使いません。

なので削除したり、再設定を一からやり直す場合も、展開した配下を削除すれば済むのでそのあたりは楽でいいですね。

```
cd /opt
sudo wget https://downloads.jboss.org/keycloak/11.0.2/keycloak-11.0.2.tar.gz
tar xfpz keycloak-11.0.2.tar.gz
```

■ keycloakの設定

動作させるための設定を行います。

設定ファイルは以下になります。

```
vi /opt/keycloak/standalone/configuration/standalone.xml
```

proxyが前段にあることを知らせるために

「http-listener」に、「proxy-address-forwarding="true"」を追記します。

```
- <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true" read-timeout="30000"/>
+ <http-listener name="default" socket-binding="http" proxy-address-forwarding="true" redirect-socket="https" enable-http2="true" read-timeout="30000"/>
```

IPアドレスを設定します。

```
<interfaces>
  <interface name="management">
    <inet-address value="\${jboss.bind.address.management: 【KeycloakサーバーのIP】 }"/>
  </interface>
  <interface name="public">
    <inet-address value="\${jboss.bind.address: 【KeycloakサーバーのIP】 }"/>
  </interface>
</interfaces>
```

■ KeyCloakを試験起動

```
sudo /opt/keycloak/bin/standalone.sh -Djboss.http.port=80
```

■ サービスとして登録

```
sudo vi /etc/systemd/system/keycloak.service

[keycloak.service]

[Unit]
Description=Jboss Application Server
After=network.target
[Service]
Type=idle
Environment=JBOSS_HOME=/opt/keycloak
JBOSS_LOG_DIR=/var/log/keycloak/ "JAVA_OPTS=-Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true"
User=root
Group=root
ExecStart=/opt/keycloak/bin/standalone.sh -Djboss.http.port=80 -b 0.0.0.0
TimeoutStartSec=600
TimeoutStopSec=600
[Install]
WantedBy=multi-user.target
```


設定ファイルの再読み込み

```
# sudo systemctl daemon-reload
```

サービスの自動起動有効化

```
# sudo systemctl enable keycloak
```

サービスの起動

```
# sudo systemctl start keycloak
```

■ Keycloak管理者作成

KeyCloakの操作を行うための管理者を作成します。

管理者は先にコマンドで作成しておきます。

```
sudo /opt/keycloak/bin/add-user-keycloak.sh -r master -u xxxx -p yyyy
```

■ NginxでのSSL化終端

今回も、前回同様フロントエンド側のnginxにてSSL化と終端を行います。

フロントエンド側のNginxでkeycloak用のConfigを作成します。nginx、およびcetbotを導入し、設定したURLでのLet's EncryptによるSSL証明書を取得済みです。

nginxによるSSL化終端およびリバースプロキシについては前号に掲載したものと同様ですので今回はKeyCloakのSSO動作を主体に扱いますので、省略します。すいません。

以下はnginxのConfig例になります。

```
server {
    server_name    key.test.com;
    listen 443 ssl;
    listen [::]:443 ssl;

    ssl_protocols TLSv1.2;
    ssl_ciphers EECDH+AESGCM:EECDH+AES;
    ssl_ecdh_curve prime256v1;
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 30m;
    ssl_certificate /etc/letsencrypt/live/drv.test.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/drv.test.com/privkey.pem;

    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    sendfile on;
    client_max_body_size 0;
    error_log /var/log/nginx/key.error.log;

    location / {
        # プロキシ先のサーバアドレスとポート番号を指定
        proxy_pass http://192.168.1.11/ ;
        proxy_buffering off;
        proxy_set_header X-Forwarded-Host $host;
```

```

proxy_set_header    X-Real-IP           $remote_addr;
proxy_set_header    X-Forwarded-Server $host;
proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header    X-Forwarded-Proto https;
# proxy_http_version 1.1;
# proxy_set_header    Upgrade $http_upgrade;
# proxy_set_header    Connection "upgrade";
# proxy_cookie_path  /guacamole/ /;
}

gzip on;
}

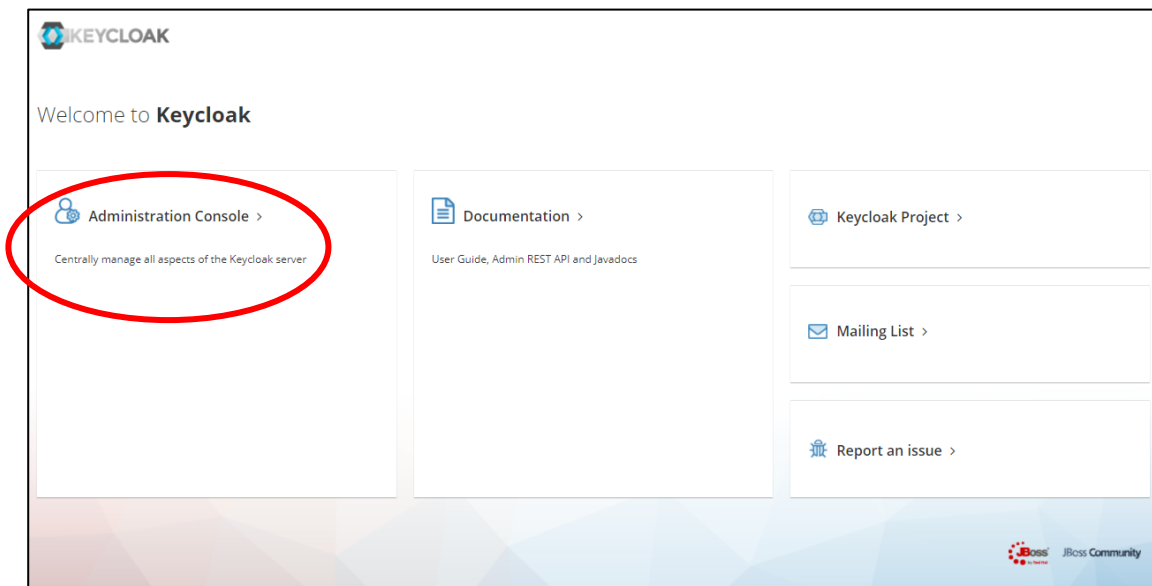
```

■ Keycloakの設定

Keycloakを起動したら下記URLにアクセスし、赤丸箇所をクリックしてログインします。

`https://[KeycloakサーバーのFQDN]/auth/`

ログインID/PWDは上記「Keycloak管理者作成」で作成したユーザーです。



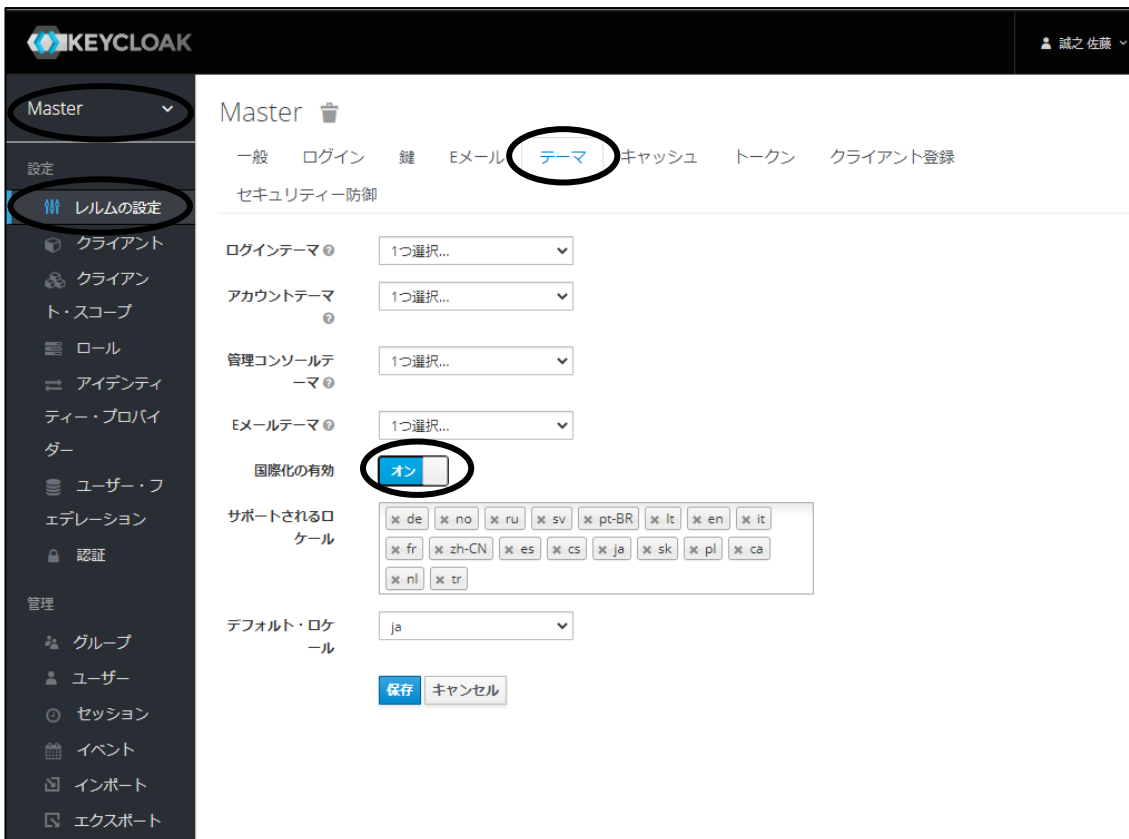
上記はすでに日本語化された状態になっていますが、日本語表示がされていない場合は、下記の赤丸の個所、英語で「Internationalization Enabled」となっている個所をONにします。

なお、画面右上の丸囲いが「Master」となっている状態で実施してください。

レルムの設定 ⇒ Realm Settings

テーマ ⇒ Themes

国際化の有効 ⇒ Internationalization Enabled

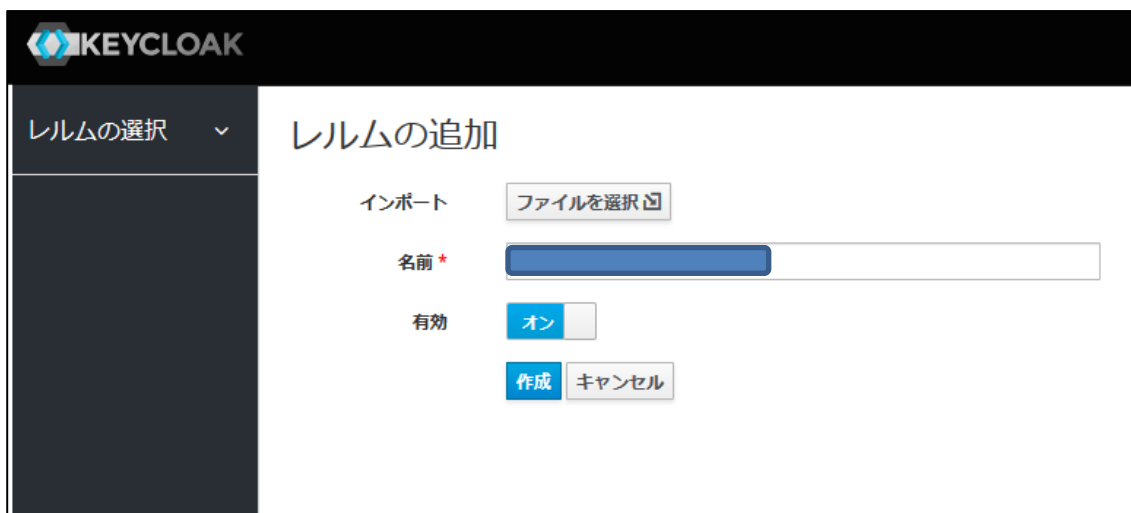


The screenshot shows the Keycloak Admin Console interface. The top navigation bar includes the Keycloak logo and the user name '誠之 佐藤'. The left sidebar shows the navigation menu with 'Master' selected at the top and 'レルムの設定' (Realm Settings) highlighted. The main content area is titled 'Master' and contains several tabs: '一般' (General), 'ログイン' (Login), '鍵' (Keys), 'Eメール' (Email), 'テーマ' (Themes), 'キャッシュ' (Cache), 'トークン' (Tokens), and 'クライアント登録' (Client Registration). The 'テーマ' tab is active and circled in red. Below the tabs, there are several settings: 'ログインテーマ' (Login Theme), 'アカウントテーマ' (Account Theme), '管理コンソールテーマ' (Admin Console Theme), and 'Eメールテーマ' (Email Theme), each with a '1つ選択...' dropdown menu. The '国際化の有効' (Internationalization Enabled) checkbox is checked and circled in red. Below this, there is a section for 'サポートされるロケール' (Supported Locales) with a grid of language codes: 'x de', 'x no', 'x ru', 'x sv', 'x pt-BR', 'x lt', 'x en', 'x it', 'x fr', 'x zh-CN', 'x es', 'x cs', 'x ja', 'x sk', 'x pl', 'x ca', 'x nl', and 'x tr'. At the bottom, there is a 'デフォルト・ロケール' (Default Locale) dropdown menu set to 'ja' and two buttons: '保存' (Save) and 'キャンセル' (Cancel).

続いて「Master」にオンマウスすると表示される「レルムの追加」をクリックします。
なお当方の実際の検証環境からすでに追加したレルムが表示されてる部分をマスクしています。



レルム名は任意のものを入力してください。



次に、認証を行うクライアントを登録します。クライアントIDとクライアントプロトコルを登録します。今回はOpenid-Connectを使用します。クライアントIDにRocketChatであることをわかるように名前を付けていますが、任意の名前で大丈夫です。

KEYCLOAK

クライアント > クライアントの追加

クライアントの追加

インポート

クライアントID*

クライアントプロトコル

ルートURL

保存を押して作成したら、必要な情報を登録します。

クライアント・プロトコル	<input type="text" value="openid-connect"/>
アクセスタイプ	<input type="text" value="confidential"/>
標準フローの有効	<input checked="" type="checkbox"/>
インプリシット・フローの有効	<input type="checkbox"/>
ダイレクト・アクセス・グラントの有効	<input checked="" type="checkbox"/>
サービス・アカウントの有効	<input type="checkbox"/>
認可の有効	<input type="checkbox"/>
ルートURL	<input type="text" value="https://chat.test.com"/>
*有効なリダイレクトURI	<input type="text" value="/_oauth/keycloak"/>

- アクセスタイプ** : Confidential
- ルートURL** : ここでは前号で構築したRocket.ChatのURLを入力しています。
- 有効なリダイレクトURL** : /_oauth/keycloak を入力します。

なお、資料やWeb記事によっては、ここにルートURLからのパスを書いているケースがあるのですが、当方環境では(URL表記ですし)ルートURLに続けて入力すべきパスを記載しないと正常にリダイレクトできませんでした。

当初、ここにもホスト名からのフルパスを入力していたところ、ルートURLに続けてこのURIが設定されてしまっていたので、この記述で正しいと思います。

設定が完了したら「保存」を押下します。

次に「クレデンシャル」タブから、Rocket.Chat側に設定する必要があるシークレットをコピーします。



前回構築したRocket.ChatはLDAPで認証しています。

ですので、KeyCloakからもLDAPで認証できるように構成します。

「ユーザー・フェデレーション」をクリックして登録していきます。



KEYCLOAK 誠之 佐藤

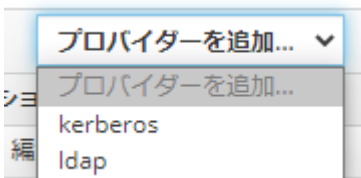
ユーザー・フェデレーション

プロバイダーを追加...

ID	有効	プロバイダー名	優先度	アクション
ldap	true	Ldap	0	編集 削除

上記はすでに登録されているので「LDAP」というのがありますが、読み替えていただければ。

「プロバイダーを追加」から、ldapを選択します。



プロバイダーを追加... ▼

- プロバイダーを追加...
- kerberos
- ldap

次ページにユーザーフェデレーションとしてのLDAPの設定を記載します。

ユーザー・フェデレーション・プロバイダーの追加

必要な設定

有効 オン

コンソール表示名

優先度

ユーザーのインポート オン

編集モード

登録の同期 オフ

* ベンダー

* ユーザー名のLDAP属性

* RDN LDAP属性

* UUID LDAP属性

* ユーザー・オブジェクト・クラス

* 接続URL

* ユーザーDN

カスタムユーザーLDAPフィルター

検索スコープ

* バインドタイプ

* バインドDN

* バインド・クレデンシャル

> Advanced Settings

> 接続プーリング

> Kerberosと統合

> 同期の設定

> キャッシュ設定

基本的にLDAPの設定は前号のLDAPで設定したものに合わせています。

ベンダー : Other

ユーザー名のLDAP属性 : cn

RDN LDAP属性 : cn

UUID LDAP属性 : entryUUID

ユーザー・オブジェクト・クラス : inetOrgPerson, organizationalPerson

接続URL : ldap:// rlyeh.local.test

※LDAPサーバのURLです。前号で構築したものを入力しています。

ユーザーDN : ou=People,dc=test,dc=com

※オンマウスすると表示されますがユーザーが登録されているLDAPの完全DNを登録します。

バインドタイプ : simple (変更しない)

バインドDN : uid=user01,cn=ldapmanager,ou=Group,dc=test,dc=com

※こちらも前号で作成したディレクトリマネージャのDNを登録します。

バインド・クレデンシャル : 上記バインドDNのパスワードを設定します。

ここまで設定したら、「認証テスト」をクリックして認証が可能な状態になっているか確認します。

■ Rocket.Chatの設定

今回はKeyCloakを用いてRocket.Chatのログインを行いますので、Rocket.Chat側にも設定が必要です。

Rocket.ChatにはデフォルトでOpenIDで認証を行う機能がありますので、そこに登録していきます。

「ログイン」⇒「管理」⇒「OAuth」に「Custom OAuth」として作成します。

今回は「Keycloak」という名称で作成しています。

次ページにスクリーンショットを掲載していますが、どこに何を入力する必要があるかを簡単に箇条書きします。

URL : `http://{keycloak_ip_address}:{port}/auth`

トークンパス : `/realms/{realm_name}/protocol/openid-connect/token`

送信されたトークン : ヘッダ

送信されたIDトークン : 「送信されたトークン」と同じ

個人情報パス : `/realms/{realm_name}/protocol/openid-connect/userinfo`

認証パス : `/realms/{realm_name}/protocol/openid-connect/auth`

範囲 : openid

アクセストークンのパラメータ名 : access_token

Id : keycloakで作成したRocket.ChatクライアントのID

パスワード : Rocket.Chatクライアントの作成時に[資格情報]で提供されるシークレットキー

ボタンテキスト : ログイン時のボタンに表示させる文言を編集できます。

なお、KeyCloakの「コールバックURI」に記載する情報はこのカスタムOAuthの冒頭に記載されているものを入力したものです。

管理 ×

OAuth アプリケーション

メーラー

権限

ログ表示

アプリ

Marketplace

設定

検索

Atlassian Crowd

Blockstack

CAS

Discussion

E2E暗号化

Federation

IRCフェデレーション

LDAP

OAuth

Rate Limiter

SAML

SlackBridge

Smarsh

SMS

WebDAV統合

WebRTC

アカウント

アセット

アナリティクス

オフコ会話

カスタムサウンドファイルシステム

カスタム絵文字ファイルシステム

スレッド

セットアップウィザード

ビデオ会議

ファイルアップロード

プッシュ通知

ポット

メタ情報

メッセージ

メール

ユーザーデータのダウンロード

ライブストリーム & ブロードキャスト

ライブチャット

レイアウト

OAuth 変更を保存 OAuthサービスをリフレッシュする カスタム認証を追加

Custom OAuth: Keycloak ^

OAuth プロバイダーを構築する場合、コールバック URL として [https://\[redacted\]/oauth/keycloak](https://[redacted]/oauth/keycloak) を提供してください。

有効にする はい いいえ 🔄

URL 🔄

トークンパス 🔄

トークン送信元 🔄

経由するIDトークン 🔄

個人情報パス 🔄

認証パス 🔄

範囲

アクセストークンのパラメータ名

ID 🔄

パスワード 🔄

ログイン方法

ボタンテキスト 🔄

ボタン文字色

ボタン背景色

ユーザー名フィールド 🔄

アバターのフィールド

ロール/グループのフィールド名

SSOからロールのマージ はい いいえ

ユーザーをマージする はい いいえ

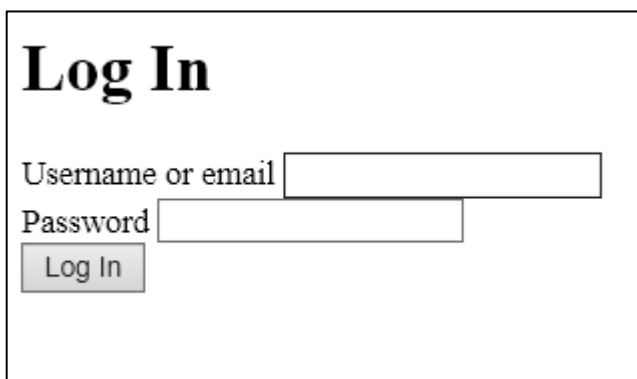
セクション設定のリセット リセット

カスタム OAuth を削除



いったんログアウトすると、ログイン画面に「KEYCLOAKでログイン」のボタンが追加されています。

今回は通常ログインも禁止していませんので、上下にボタンが表示される状態になっています。ここで「KEYCLOAKでログイン」を押下するとKeyCloakのログイン画面が表示されますので、LDAPに登録しているユーザーネームとパスワードでログインができます。



今回は外観などは全くいじってないので上記のように殺風景なログインウィンドウが出ます。

ユーザー名orEmailとパスワードでログインができます。

注意点としては、別IDでも同一Emailが設定されているユーザーがいたり、ローカルで同じアカウント名が登録されていると「account already exists」というメッセージが出る場合があります。また、頻繁にログイン/ログアウトを繰り返していると、DB側の遅延の影響でログインに失敗することもあるようなので、その点は注意が必要です。

■おわりに

今回の設定でKeyCloakログインを行うと、以降ログインせずともRocket.Chatにログインできます。もちろんこれは本来他のシステムにも適用することを目的にしているのですが、1つのシステムだけ適用してもあまり意味はないんですが、まずはサワリということで。。

とはいえ認証系のOSSは、種類が少ないということもありますが、それよりも認証のメカニズムや設定の方法などについてきちんと理解をしたうえで設定しないと、ログインできなくなったり逆にセキュリティ的に問題のある状態になることもあるので、そのあたりは注意しましょう。

■あとがき

今回はVol2に引き続きの形で、Extra Edition 2として発行しました。

今後の企画ですが、

「仮想的な社内の全システムをOSSで構築する」

というのを演習がてら1年くらいかけてやっていきたいと考えています。

前回導入したものや、今回のKeyCloak、まだ記事化していませんがZabbixなどの監視、SNIPE ITなどのアセット管理、その他ログ監査やメール監査なども取りまとめたシステムまるまる一式を全部OSSでくみ上げられるか？とだいぶ壮大なことを考えていたりします。

1つ1つはそれほどヘビーなものでもなくとも、仮想的なものとは言え実際に使えるような状態に組むとなるとかなりの規模のものになると思うので、どこまでできるかはわかりませんが、まとめられそうなら徐々に記事にしていければと思います。

お目通しいただき、ありがとうございました。

編集/発行/文責 辻瀬蒼伊

「dabble in...」 Extra Edition 2

発行日 : 2020/12/26 技術書典10 発行

発行 : Far Northern Other World(Fnow)

〒107-0052

東京都港区赤坂 1 - 4 - 4 富士野ビル501

辻瀬蒼伊(立神梢一) aka 佐藤誠之

URL : <https://www.fnwo.org>

メール : makoyuki@fnwo.org

Copyright © 2020 Seiji Satou All Rights Reserved

D A B B L E I N . . .

E X T R A E D I T I O N 2

F A R N O R T H E R N O T H E R W O R L D