Auth0 IdP を利用した Qlik Sense Enterprise on Kubernetes の設定

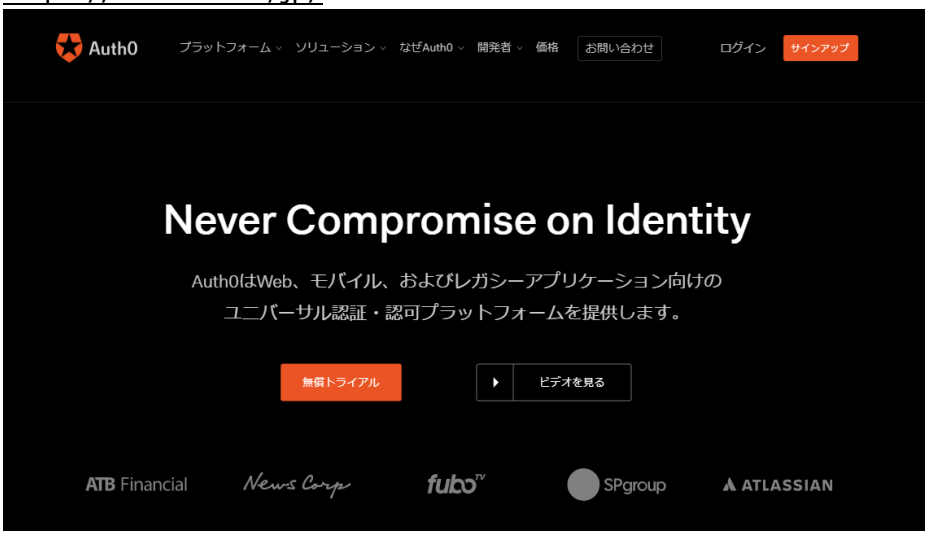
Qlik Sense Enterprise マルチクラウド演習

目次

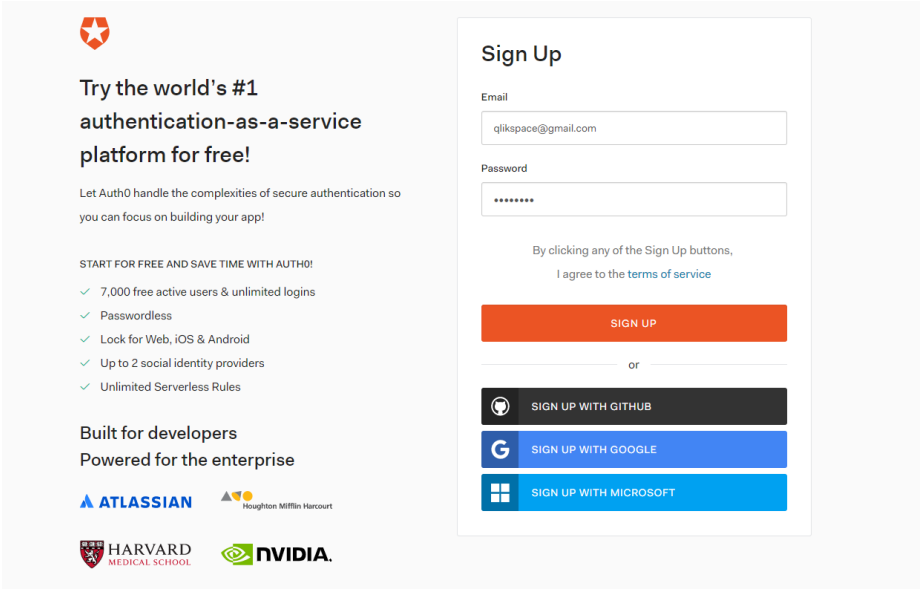
1.Auth0 へのアカウント登録

1) 以下のサイトにアクセスし、「無償トライアル」をクリックします。

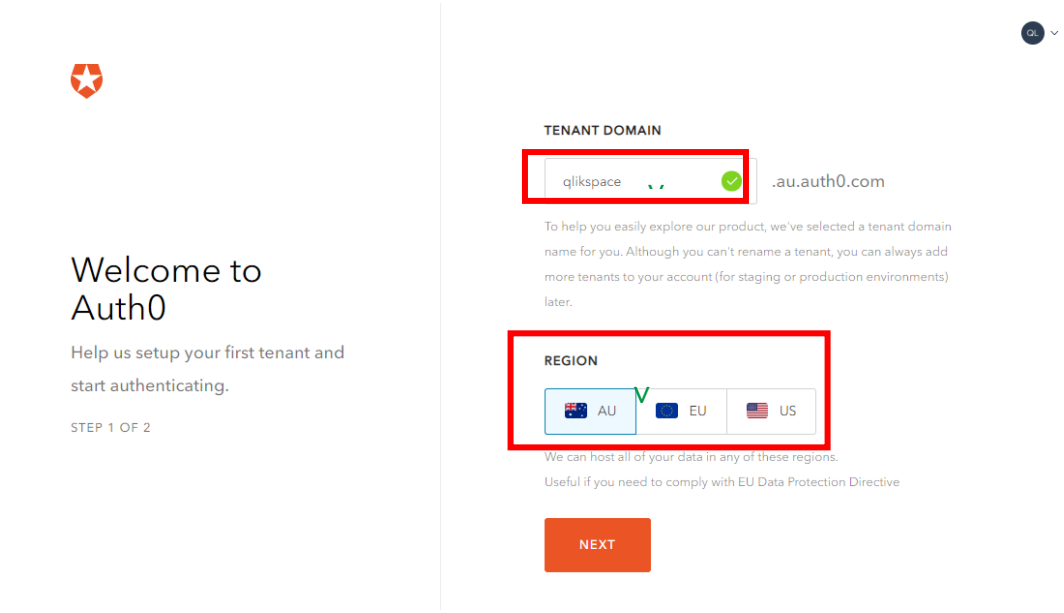
https://auth0.com/jp/



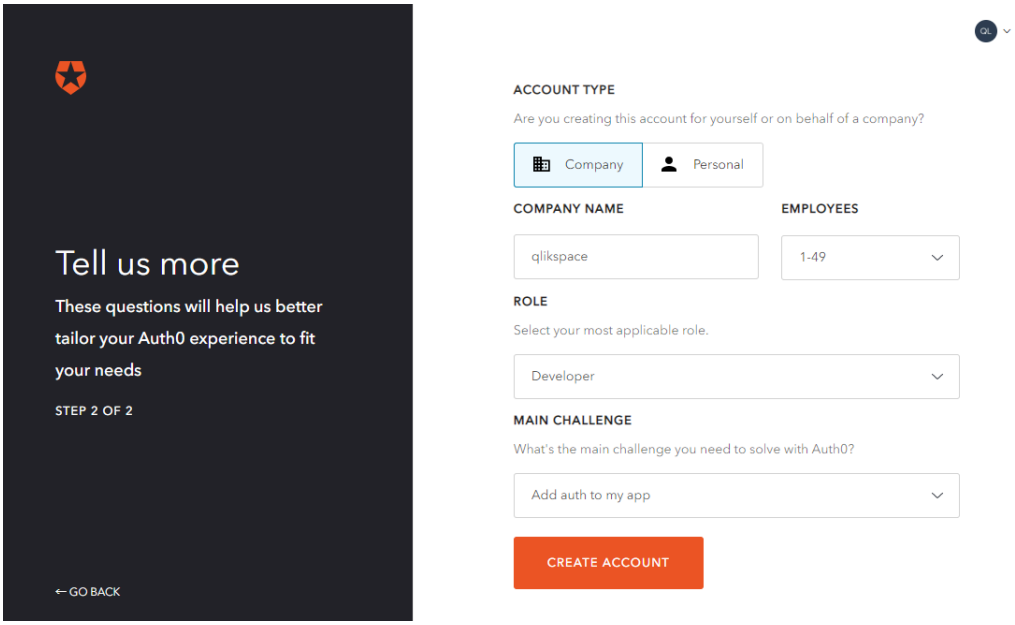
2) メール、Github、Google もしくは Microsoft のアカウントでサインアップします。



3) 任意の Tenant Domain を入力し、Region を選択して「Next」をクリックします。

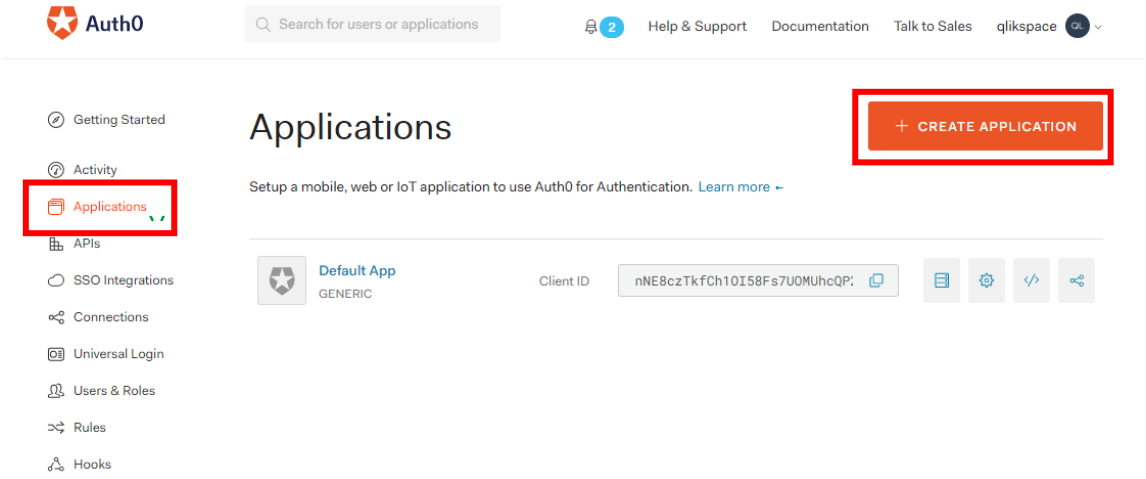


4) 項目を任意に入力し、「Create Account」をクリックします。

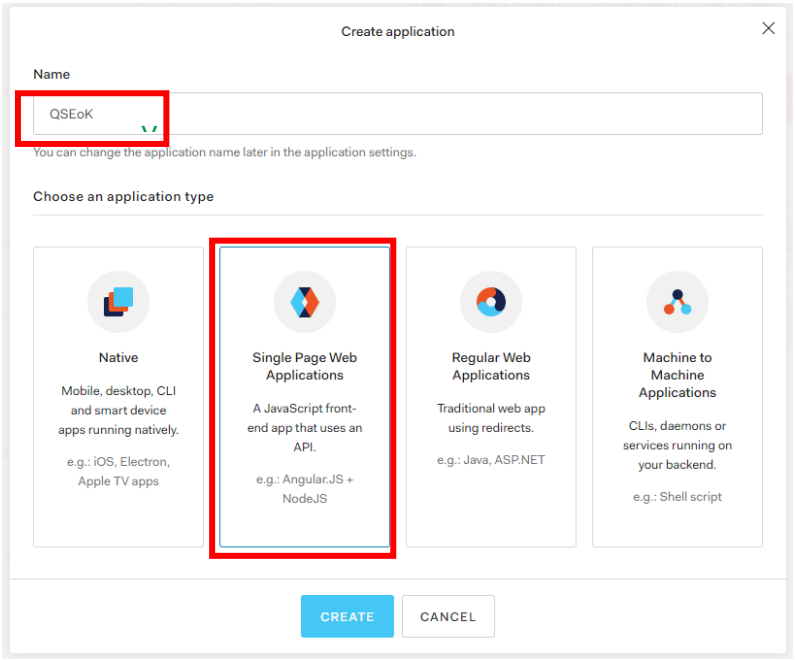


2.Auth0 アプリケーションの作成

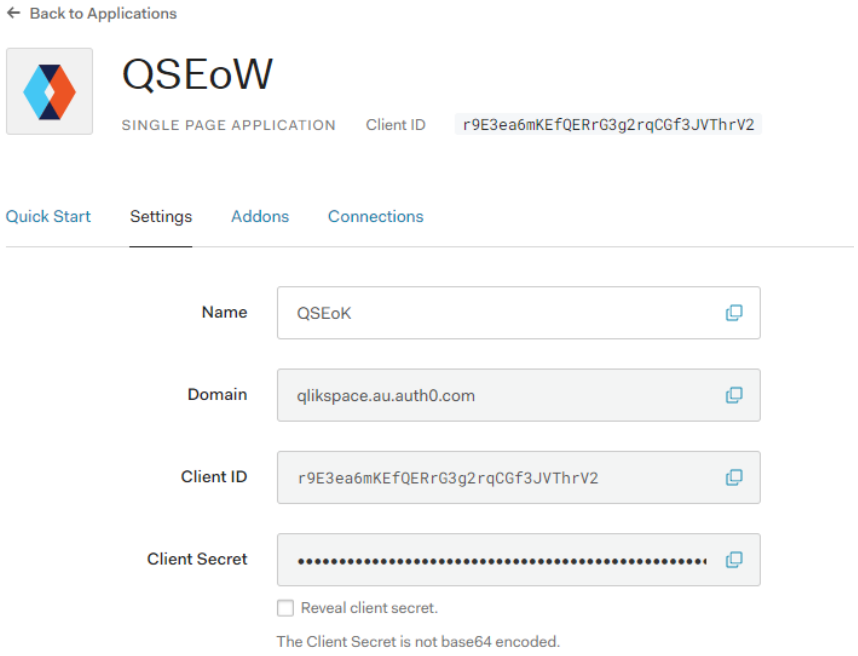
1) 左側のメニューから「Applications」を選択し、「Create Application」をクリックします。



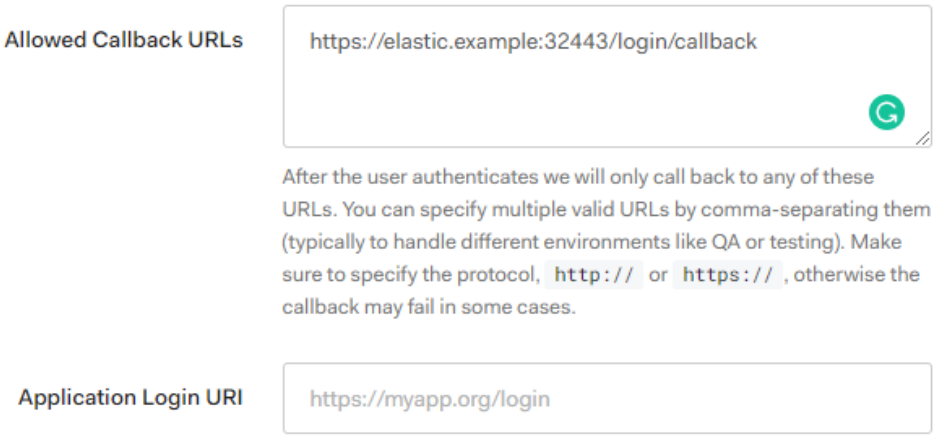
2) 「Name」に「QSEoK」と入力し、「Single Page Web Applications」を選択して「Create」をクリックします。



3) 「Settings」タブを開き、後続のステップで利用しますので「Domain」「Client ID」「Client Secret」をメモに残してください。



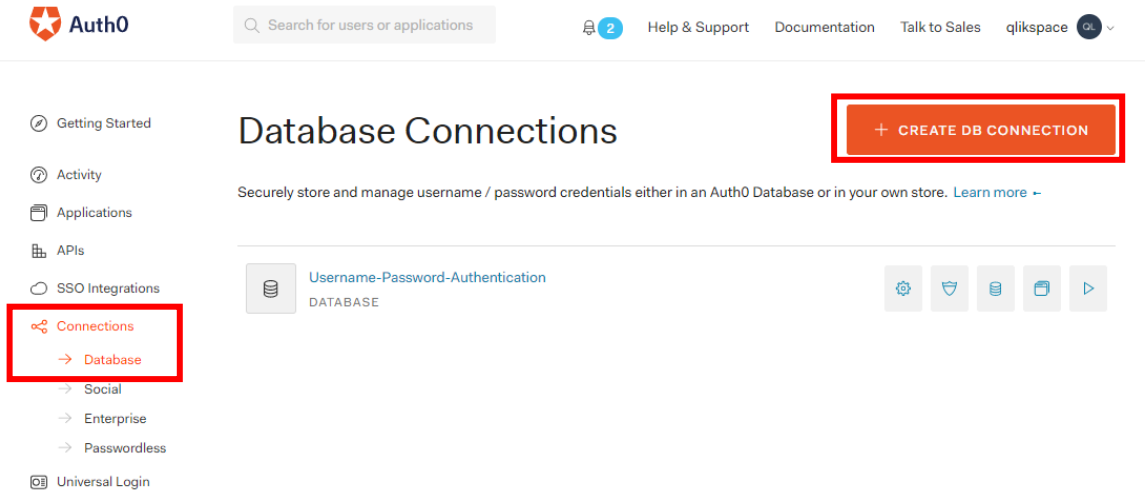
4) 下にスクロールし、「Allowed Callback URLs」に「https://elastic.example:32443/login/callback」を入力します。(Azure の場合にはポート番号は不要です。また AWS では取得した URL に /login/callback を付加した URL をここで指定します。) Auth0 では登録された URL にのみコールバックを行い、ここでは Qlik Sense Enterprise on Kubernetes の URL に/login/callback を付加した URL を入力します。



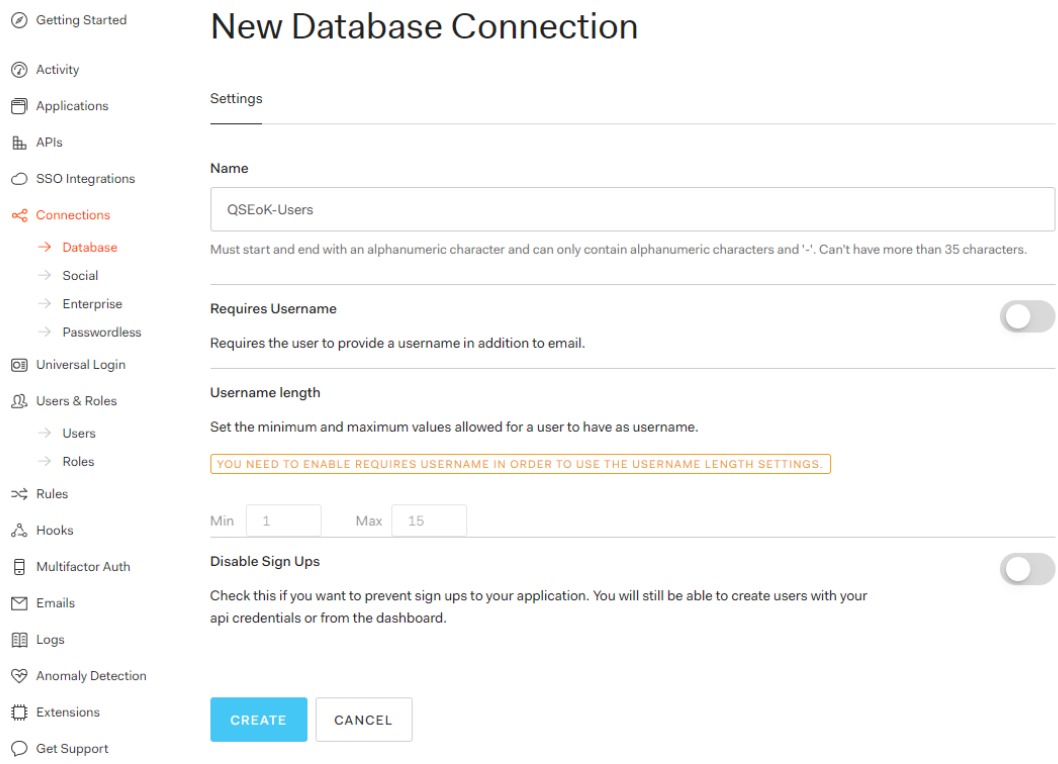
5) 下までスクロールして「Save Changes」をクリックします。

3.ユーザーデータベースの設定とユーザーの登録

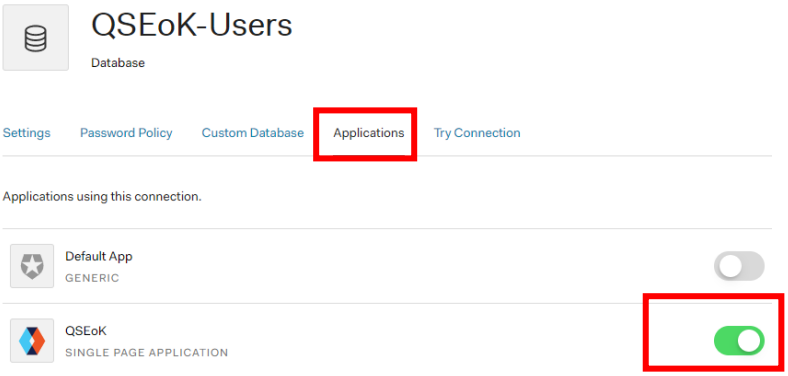
1) 左側メニューから「Connections」>「Database」を選択し、「Create DB Connection」をクリックします。



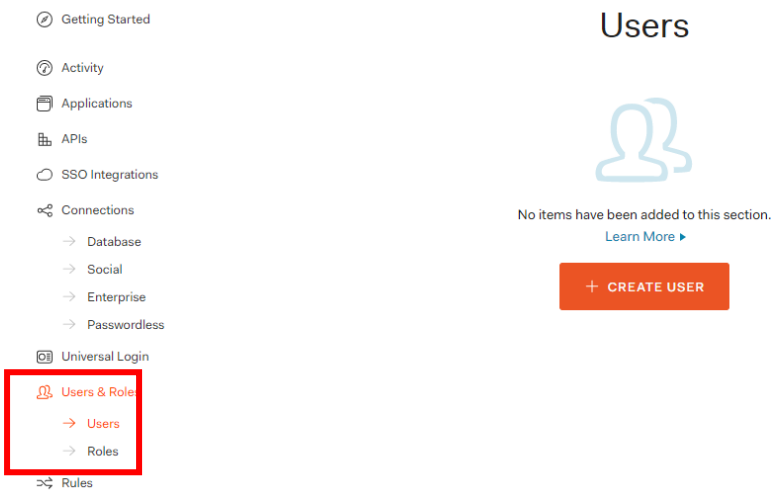
2) 「Name」に「QSEoK-Users」と入力し「Create」をクリックします。



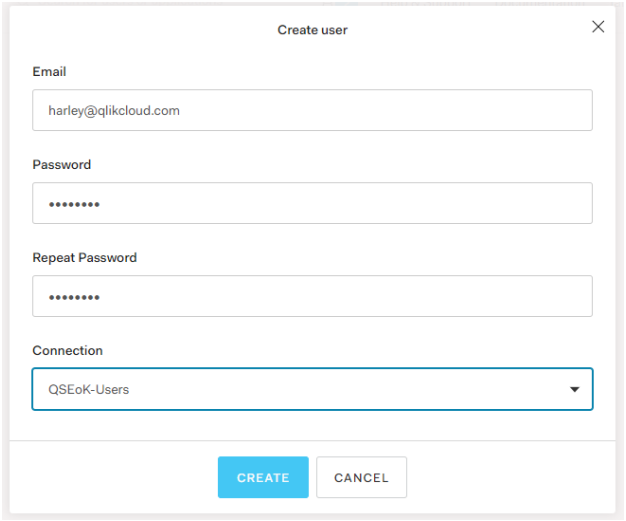
3) 「Application」タブを開き、先ほど作成した「QSEoK」のアプリケーションのスイッチを有効化します。



4) 左側メニューから「Users & Roles」>「Users」を選択し、「Create User」をクリックします。



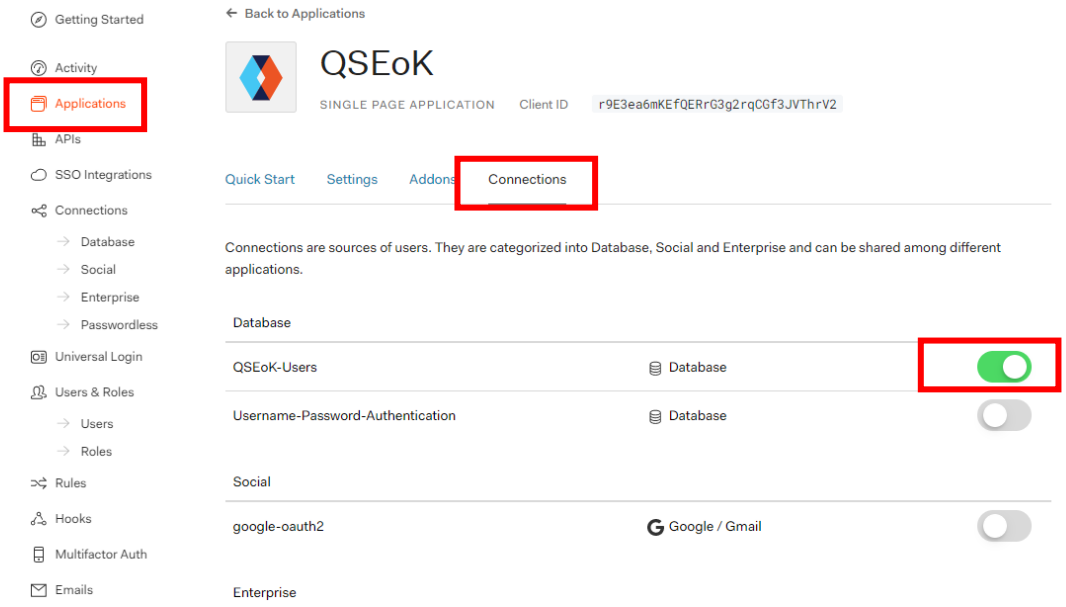
5) ユーザー情報を入力します。(メールは実在しないメールでも問題ありません)「Connection」には先ほど作成した「QSEoK-Users」を選択して「Create」をクリックします。



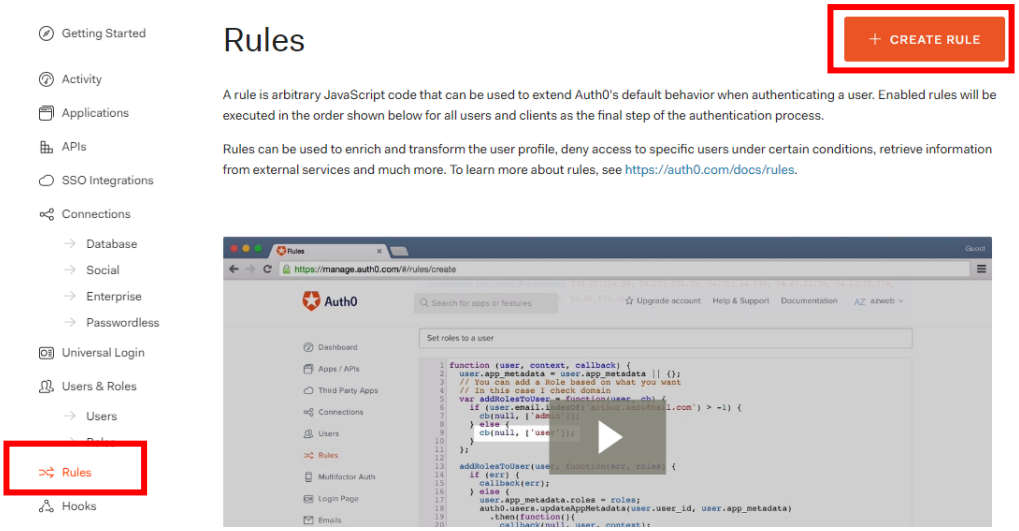
6) このユーザーにグループの設定を行います。スクロールし、「user_metadata」に以下の入力を行い「Save」をクリックします。他のユーザーを作成する場合には同様の作業を繰り返します。

```
{
"groups": [
"Everyone"
]
}
```

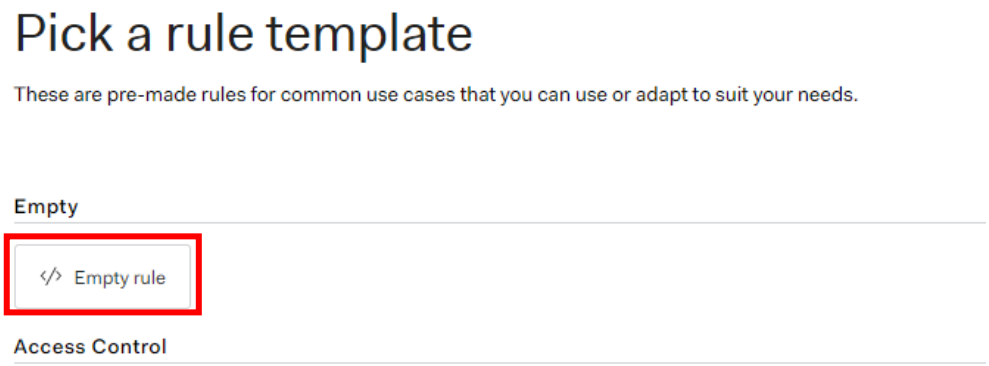
7) 左側メニューから「Applications」を選択して先ほど作成した「QSEoK」のアプリケーションを選択し、「Connections」タブをクリックして作成した「QSEoE-Users」の Database のみを有効化して他は無効化します。



8) 左側メニューから「Rules」を選択して「Create Rule」をクリックします。

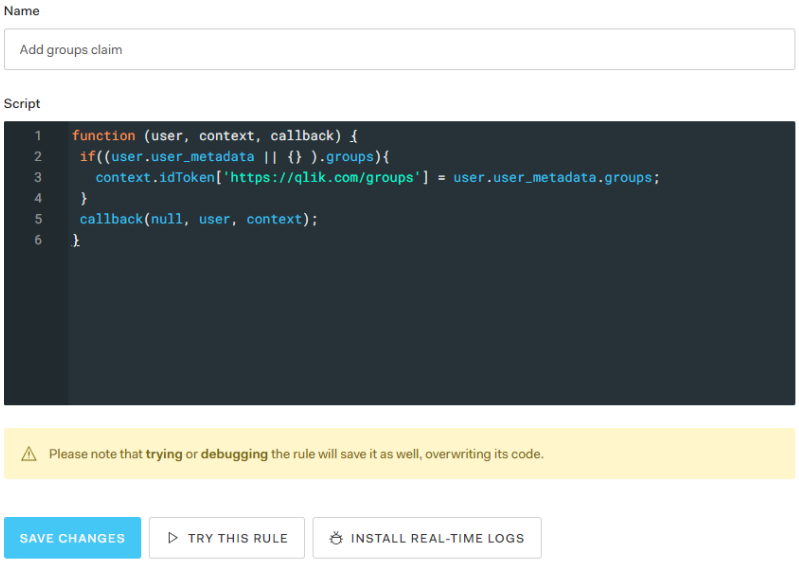


9) 「Empty rule」を選択します。



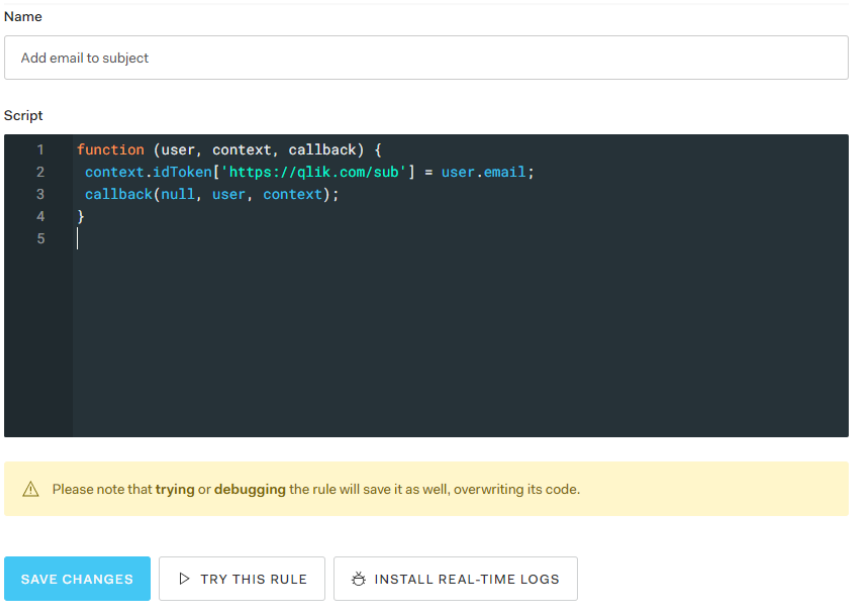
10)　「Script」を以下の通り変更し、「Name」を「Add groups claim」に変更して「Save Changes」をクリックします。これにより認証に Auth0 から Qlik Sense Enterprise に渡されるメタデータにグループ情報が含まれます。

```
function (user, context, callback) {
 if((user.user_metadata || {} ).groups){
   context.idToken['https://qlik.com/groups'] = user.user_metadata.groups;
 }
 callback(null, user, context);
}
```

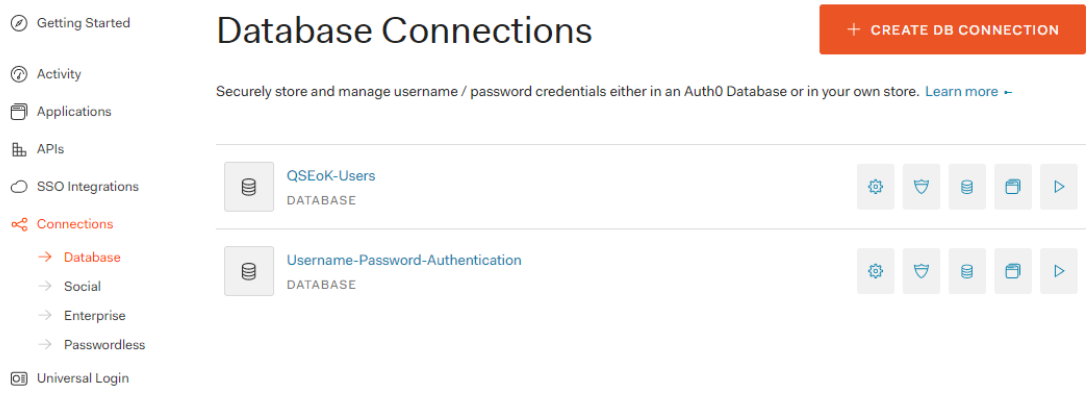


11)　同様の手順でもう一つ別の Empty rule を作成し、「Script」を以下の通り変更して「Name」を「Add email to subject」に変更し、「Save Changes」をクリックします。これにより文字列で表現される User ID を理解しやすい email で置き換えて表現します。

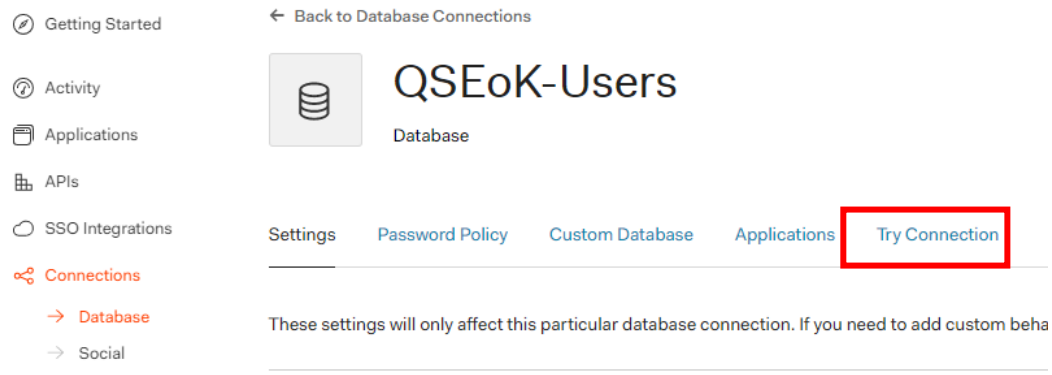
```
function (user, context, callback) {
 context.idToken['https://qlik.com/sub'] = user.email;
 callback(null, user, context);
}
```



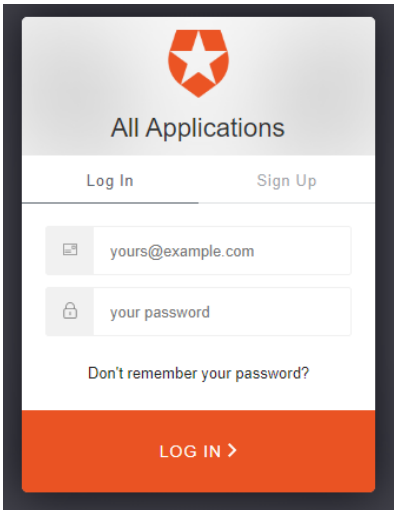
12) ユーザーログオンのテストを行います。「Connections」>「Database」を選択して、先ほど作成した「QSEoK-Users」をクリックします。



13) 「Try Connection」をクリックします。



14)　ログオン画面が表示されますので、先ほど追加したユーザーの Email とパスワードを入力します。



15)　ログオンが成功し、メタデータに追加したルールに従って groups に Everyone、sub に Email が含まれていることを確認します。

4.Qlik Sense Enterprise on Kubernetes 側（Minikube を利用の場合）での Auth0 を使った認証の設定

ここでは Minikube を使った場合の Qlik Sense Enterprise on Kubernetes 側の設定方法をご説明します。Azure Kubernetes Service（AKS)や Amazon Elastic Kubernetes Service (AKS)などの設定を行う場合にはそれぞれのドキュメントをご参照ください。

1) この文書では以降の導入作業を root ユーザーで実施します。Windows ホスト上で PowerShell を開き、コンソールから以下で SSH アクセスを行います。パスワードには「osboxes.org」を入力します。

```
ssh root@elastic.example
```

2) エディタを利用して以下の内容の values-auth0.yaml ファイルを作成します。尚、以下は Minikube を利用した場合の例になりますので、Azure Kubernetes Service(AKS)や Amazon Elastic Kubernetes Service(EKS)などを利用するなどの場合には別紙をご参照の上、そこで指定された YAML ファイルを利用してください。

前段の演習「01_QSE on Kubernetes の導入」で NFS Provisioner の設定を既に行っているか、いないかで利用すべきファイルの内容が異なります。(どちらを利用すべきか不明な場合は「① NFS Provisioner の設定を行っていない場合」をご利用ください。) また、黄色でハイライトされている「discoveryUrl」、「clientId」、「clientSecret」、「postLogoutRedirectUri」の値は前段の手順でメモした Auth0 上のアプリケーションの「Domain」、「Client ID」、「Client Secret」の値に基づいて書き換える必要があります。

① NFS Provisioner の設定を行っていない場合

```
#This setting enables dev mode to include a local MongoDB install
devMode:
  enabled: true

#This setting accepts the EULA for the product
engine:
  acceptEULA: "yes"

# MINIKUBE SPECIFIC SETTINGS (dont not use with other K8
providers)_____________
elastic-infra:
 nginx-ingress:
   controller:
     service:
       type: NodePort
       nodePorts:
         https: 32443
     extraArgs.report-node-internal-ip-address: ""

hub:
 ingress:
```

```
  annotations:
    nginx.ingress.kubernetes.io/auth-signin:
https://$host:32443/login?returnto=$request_uri

management-console:
 ingress:
  annotations:
    nginx.ingress.kubernetes.io/auth-signin:
https://$host:32443/login?returnto=$request_uri

edge-auth:
  oidc:
    redirectUri: https://elastic.example:32443/login/callback

identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://<Domain>/.well-known/openid-configuration"
        clientId: "r9E3ea6mKEfQERrG3g2rqCGf3JVThrV2"
        clientSecret :
"BZLGoB5ubfI1l2mwBVfxUkX654QiEJTiaXOGEtcTxr7CO8D5yNVFYHWp6Tm5zD1g"
        realm: "Auth0"
        postLogoutRedirectUri: "https://<Domain>/v2/logout/"
        hostname: "elastic.example"
        claimsMapping:
          client_id: [ "client_id", "azp" ]
          groups: "/https:~1~1qlik.com~1groups"
          sub: ["/https:~1~1qlik.com~1sub","sub"]
```

② NFS Provisioner の設定を行っている場合

```
#This setting enables dev mode to include a local MongoDB install
devMode:
  enabled: true

#This setting accepts the EULA for the product
engine:
  acceptEULA: "yes"

#These setting specifies the storage for the services, in this case, NFS
```

```
global:
  persistence:
    storageClass: nfs-client

#MongoDB is set to persist with NFS storage so that data will not be lost if the
container is destroyed
mongodb:
  persistence:
    enabled: true
    accessMode: ReadWriteMany
    storageClass: nfs-client

# MINIKUBE SPECIFIC SETTINGS (dont not use with other K8
providers)_____
elastic-infra:
 nginx-ingress:
   controller:
     service:
       type: NodePort
       nodePorts:
         https: 32443
     extraArgs.report-node-internal-ip-address: ""

hub:
 ingress:
   annotations:
     nginx.ingress.kubernetes.io/auth-signin:
https://$host:32443/login?returnto=$request_uri

management-console:
 ingress:
   annotations:
     nginx.ingress.kubernetes.io/auth-signin:
https://$host:32443/login?returnto=$request_uri

edge-auth:
  oidc:
    redirectUri: https://elastic.example:32443/login/callback
```

```
identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://<Domain>/.well-known/openid-configuration"
        clientId: "r9E3ea6mKEfQERrG3g2rqCGf3JVThrV2"
        clientSecret :
"BZLGoB5ubfI1l2mwBVfxUkX654QiEJTiaXOGEtcTxr7CO8D5yNVFYHWp6Tm5zD1g"
        realm: "Auth0"
        postLogoutRedirectUri: "https://<Domain>/v2/logout/"
        hostname: "elastic.example"
        claimsMapping:
          client_id: [ "client_id", "azp" ]
          groups: "/https:~1~1qlik.com~1groups"
          sub: ["/https:~1~1qlik.com~1sub","sub"]
```

3) Qlik Sense Enterprise on Kubernetes を展開します。

```
helm upgrade --install qliksense qlik/qliksense -f values-auth0.yaml
```

4) 以下を実行し、展開した Pod の状況を確認します。STATUS が ContainerCreating から全て Running となるまで待ちます。

```
kubectl get pods
```

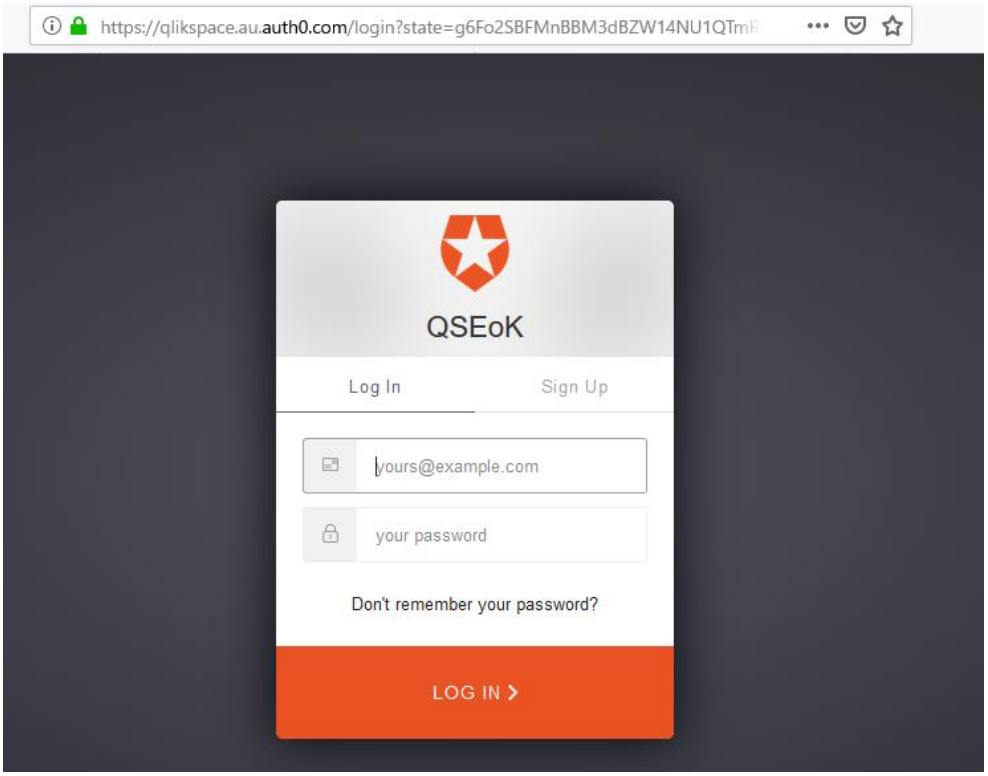
以下のコマンドを実行すると定期的な繰り返し処理で上記のコマンドを実行することも可能です。

```
while [ 1 ]; do kubectl get pods; sleep 5; clear; done
```

5) Windows のホストコンピューター上でブラウザを開き、以下の URL にアクセスします。

https://elastic.example:32443/

6) Auth0 のログイン画面が表示されますので、先ほど追加したユーザーの Email とパスワードを入力します。



7) 以下のようなハブのトップ画面が表示されます。これで環境のセットアップが完了しましたので、別紙「03_Qlik Sense Enterprise on Kubernetes 上の管理コンソールでのライセンス有効化」の手順に従ってサイトのライセンスを有効化します。

5.認証エラーの場合の Qlik Sense Enterprise on Kubernetes 側のログの確認

1) 以下のコマンドを実行し、edge-auth の Pod の ID を確認します。

```
kubectl get pods | grep auth
```

2) 以下のコマンドで edge-auth Pod の edge-auth コンテナのログを表示します。(必要に応じて grep で絞り込みを行います。)

```
kubectl logs qliksense-edge-auth-78f4f76cb8-722d9 edge-auth
```

※ 以下のように-f フラグを追加するとストリーム表示となります。

```
kubectl logs qliksense-edge-auth-78f4f76cb8-722d9 edge-auth -f
```

3) 以下のようなログが表示され、ブラウザからの URL リクエストやエラーの内容が表示されます。


```
{"ip":"172.17.0.1","url":"/login/callback?code=***MASKED***&state=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyZnAiOiJVQWJpMU1SekItemtYTGdFeWh0LTdKOEhtNVFaVlk1ZFJSeG5seFRGMlNBIiwicmQiOiJodHRwczovL2VsYXN0aWMuZXhhbXBsZTozMjQ0My8iLCJpYXQiOjE1NjcwMDExNDksImV4cCI6MTU2NzYwNTk0OX0.Amna7F8boR8r11RNq0fPssTJVyE72OlfEIzB6D2GcT0","method":"GET","query":[["code","***MASKED***"],["state","eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyZnAiOiJVQWJpMU1SekItemtYTGdFeWh0LTdKOEhtNVFaVlk1ZFJSeG5seFRGMlNBIiwicmQiOiJodHRwczovL2VsYXN0aWMuZXhhbXBsZTozMjQ0My8iLCJpYXQiOjE1NjcwMDExNDksImV4cCI6MTU2NzYwNTk0OX0.Amna7F8boR8r11RNq0fPssTJVyE72OlfEIzB6D2GcT0"]],"pid":1,"type":"edge-auth","version":"2.38.4","buildInfo":{"SHA":"d2953be","name":"edge-auth","version":"2.38.4","buildTime":"2019-06-21T19:13:26.093Z"},"module":"services/identityProvider","error":"AssertionError [ERR_ASSERTION]: id_token issued in the future","stack":"AssertionError [ERR_ASSERTION]: id_token issued in the future¥n    at Client.validateIdToken (/usr/src/app/node_modules/openid-client/lib/client.js:586:7)¥n    at grant.then.then.tokenset (/usr/src/app/node_modules/openid-client/lib/client.js:423:32)¥n    at process._tickCallback (internal/process/next_tick.js:68:7)","code":"EAS-7","logTraceId":"qCa0rzS8mNtZncBjmbS1p5XE","headers":{"host":"elastic.example:32443","x-request-id":"ef98a974f98691d380b9df096615c03c","x-real-
```

ip":"172.17.0.1","x-forwarded-for":"172.17.0.1","x-forwarded-host":"elastic.example:32443","x-forwarded-port":"443","x-forwarded-proto":"https","x-original-uri":"/login/callback?code=***MASKED***&state=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyZnAiOiJVQWJpMU1SekItemtYTGdFeWh0LTdKOEhtNVFaVlk1ZFJSeG5seFRGMlNBIiwicmQiOiJodHRwczovL2VsYXN0aWMuZXhhbXBsZTozMjQ0My8iLCJpYXQiOjE1NjcwMDExNDksImV4cCI6MTU2NzYwNTk0OX0.Amna7F8boR8r11RNq0fPssTJVyE72OlfEIzB6D2GcT0","x-scheme":"https","user-agent":"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:68.0) Gecko/20100101 Firefox/68.0","accept":"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8","accept-language":"en-US,en;q=0.5","accept-encoding":"gzip, deflate, br","upgrade-insecure-requests":"1"},"logseverity":"ERROR","level":"error","message":"Failure during IDP callback: AssertionError [ERR_ASSERTION]: id_token issued in the future","timestamp":"2019-08-28T14:06:20.941Z

About Qlik

Qlik is on a mission to create a data-literate world, where everyone can use data to solve their most challenging problems. Only Qlik's end-to-end data management and analytics platform brings together all of an organization's data from any source, enabling people at any skill level to use their curiosity to uncover new insights. Companies use Qlik products to see more deeply into customer behavior, reinvent business processes, discover new revenue streams, and balance risk and reward. Qlik does business in more than 100 countries and serves over 48,000 customers around the world.

qlik.com