

Hvad er SSO:

SSO er authenticationsteknologi, der giver brugere mulighed for at logge ind engang og derved få adgang til systemet.

Jeg undersøgte hvilke SSO-løsninger, der findes på markedet, og hvilke der er populære. Jeg kom frem til, at Auth0, Okta, Firebase Authentication og OneLogin er blandt de mest anvendte. Herefter begyndte jeg at undersøge mulighederne med fokus på brugervenlighed og opsætning.

<https://www.joinsecret.com/compare/auth0-vs-okta>

Alle SSO'er tilbyder en free plan.

Det er værd at bemærke at Okta opkøbte Auth0 i 2021.

### Auth0

(+) Enkel integration med mange SDK'er og eksempler.

(+) Understøtter mange sociale logins: Brugere kan logge ind med Google, Facebook, Microsoft, osv.

(+) Tilbyder 7.500 aktive brugere

(-) Pris: Kan blive dyrt for større projekter og avancerede brugsscenarier.

(-) Afhænger af Auth0's opetid og sikkerhed.

### Okta

<https://www.okta.com/pricing/#customer-identity-pricing>

inde på oktas hjemmeside, der bliver man henvist hen til Auth0, hvis man vil prøve den gratis version.

(+) Flere IAM (Identity and Access Management) muligheder - tilbyder avancerede funktioner til bruger og adgangsstyring.

(+) Kan håndtere store mængder brugere og komplekse behov.

(-) Pris: Kan være dyrt for mindre virksomheder.

(-) Komplexitet

### Firebase

(+) Enkel opsætning og god dokumentation.

(+) Gratis plan: Generøs gratis plan, ideel for mindre projekter og startups.

(+) God til projekter, der allerede bruger Firebase-tjenester.

(-) Google-afhængighed

## OneLogin

(+) Nem integration

(+) Generelt konkurrencedygtig pris sammenlignet med Okta og Auth0.

(-) Forvirrende dokumentation.

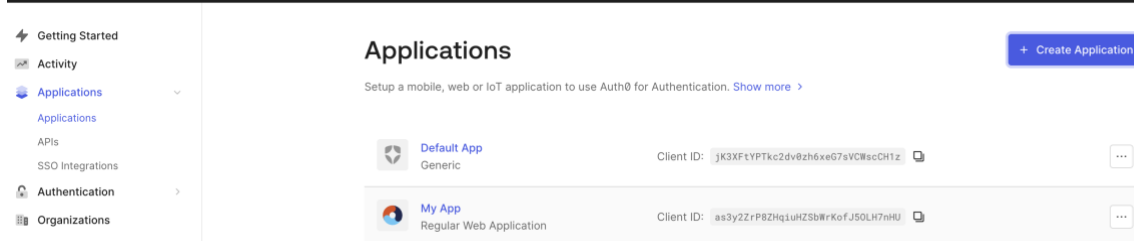
(-) Mangler nogle af de mere avancerede sikkerhedsfunktioner

Efter nøje overvejelse af fordele og ulemper ved hver løsning valgte jeg Auth0. Hovedårsagen var, at Auth0 er nemt og brugervenligt at sætte op, hvilket var en vigtig faktor for mig. Auth0 tilbyder også en generøs gratis plan, som er ideel til at komme i gang uden betydelige omkostninger. Desuden understøtter Auth0 mange sociale logins.

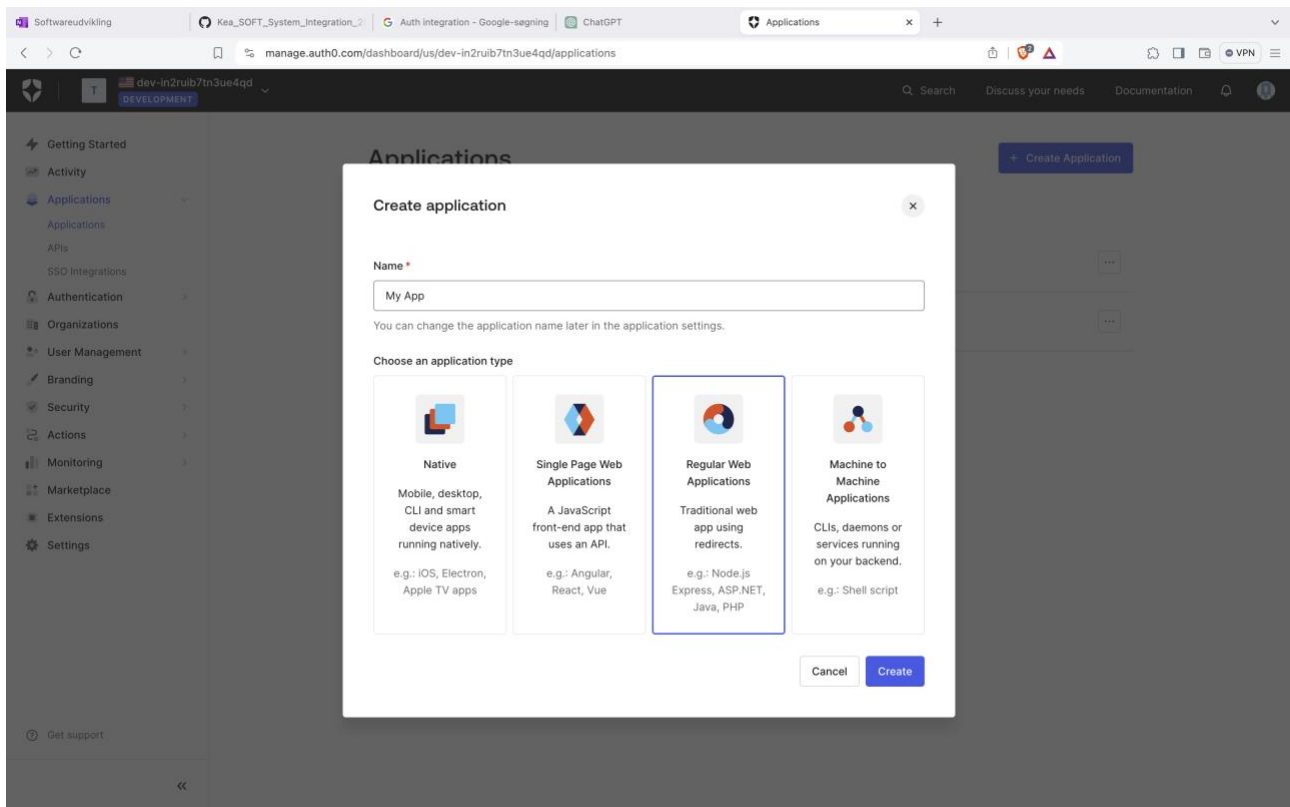
## 08c. Auth Integration:

Først opretter jeg mig inde på deres hjemmeside, hvor jeg logger ind på med min GitHub.

Efter dette går jeg ind på applikations og klikker på ”+ Create Application”



En form kommer op, og her vælger jeg et navn og vælger Regular Webb Applications, da jeg bruger Node.js med Express.



Et ny side kommer frem med et søgefelt og her skal der søges efter node.js(express), når dette er gjort, kommer der er ny form op hvor man kan vælge i mellem ”integrate with my app” eller

”explore a sample app”, og her skal der vælges ”integrate with my app”

[← Back to Applications](#)



## My App

Regular Web Application

Client ID `ip80xzamZcWyyynnMxtuXmAKM2SwZo00V`

[Quickstart](#)

[Settings](#)

[Credentials](#)

[Addons](#)

[Connections](#)

[Organizations](#)

REGULAR WEB APP / EXPRESS

# Choose your path



## I want to integrate with my app

Follow a step-by-step guide that configures Auth0 and provides code snippets to integrate into your existing app.

[Integrate Now](#)



## I want to explore a sample app

Get a pre-configured example with your account settings or check out the repository on Github.

[Explore Sample App](#)

or [View on Github](#)

Når ”integrate now” er valgt, så kommer du frem på forsiden som ser sådan her ud:

⚡ Getting Started

📄 Activity

🔗 Applications

APIs

SSO Integrations

👤 Authentication

🏢 Organizations

👥 User Management

🎨 Branding

🔒 Security

⚙️ Actions

📊 Monitoring

🛒 Marketplace

🔌 Extensions

⚙️ Settings

🔍 Get support

My App

Regular Web Application

Client ID `ip80xzamZcWyyynnMxtuXmAKM2SwZo00V`

Quickstart

Settings

Credentials

Addons

Connections

Organizations

← Change technology

Node.js (Express)

Configure Auth0

To communicate with Auth0 services you need to set up allowed URLs for specific actions on your Auth0 application. We've provided defaults so you can test in on your local environment but you can change them to match your setup.

Allowed Callback URL

A URL in your application where Auth0 redirects the user after they have authenticated.

Specify multiple valid URLs by comma-separating them. Make sure to specify the protocol `https://`, otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://`.

Allowed Logout URLs

A URL in your application that Auth0 can return to after the user has been logged out of the authorization server. This is specified in the `returnTo` query parameter.

• Configure Auth0

• Integrate the SDK

• Test your login

• Get the user profile

• Next Steps

Having trouble?

[Download a sample app](#)

[View the sample app code](#)

[Read the documentation](#)

[Ask the community](#)

[Give us feedback](#)

Denne her forsiden er en ”quickstart”, for hvordan Auth0 kan sættes op. Det første step, er at configure Auth0, som gør at vi kan kommunikere med auth0 servcies, og derfor skal vi sætte

vores allowed URL op til det dette. Auth0 default URL er localhost:3000, det kan ændres, men dette har jeg ikke gjort i mit eksempel.

#### Allowed Callback URL

A URL in your application where Auth0 redirects the user after they have authenticated.

`http://localhost:3000/callback`

Specify multiple valid URLs by comma-separating them. Make sure to specify the protocol `https://`, otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://`.

#### Allowed Logout URLs

A URL in your application that Auth0 can return to after the user has been logged out of the authorization server. This is specified in the `returnTo` query parameter.

`http://localhost:3000`

Specify multiple valid URLs by comma-separating them. Query strings and hash information are not taken into account when validating these URLs. [Learn more](#) ↗

Save Settings And Continue

Næste step er at integrere med Auth0 Services og dette gøres ved at bruge denne kommand:

## Integrate Auth0

### Install dependencies

Your application will need the `express-openid-connect` package which is an Auth0-maintained OIDC-compliant library for Express.

```
npm install express express-openid-connect --save
```

Denne pakke der er blevet installeret skal bruges til integrationen, da pakken indeholder en auth-router, der gør muligt at tilføje authentication til ens applikation. Hernæst kopieres koden som Auth0 har givet, og jeg har i mit eksempel omstruktureret det lidt, fordi jeg har brugt ECMA script. Så her kommer importen til at se således ud:

```
Import { auth } from 'express-openid-connect'
```

Herudover har jeg også sat mine configuration keys ind i en .env for ikke at offentliggøre dataen, da det er bedst practice.

## Configure Router

The Express OpenID Connect library provides the `auth` router in order to attach authentication routes to your application. You will need to configure the router with the following configuration keys:

- `baseUrl` - The URL where the application is served
- `secret` - A long, random string
- `issuerBaseUrl` - The Domain as a secure URL found in your Application settings
- `clientId` - The Client ID found in your Application settings

Here's an example already configured with your information.

```
const { auth } = require('express-openid-connect');

const config = {
  authRequired: false,
  auth0Logout: true,
  secret: 'a long, randomly-generated string stored in env',
  baseUrl: 'http://localhost:3000',
  clientId: 'ip80xzamZcWynnMxtuXmAKM2SwZo00V',
  issuerBaseUrl: 'https://dev-in2ruib7tn3ue4qd.us.auth0.com'
};

// auth router attaches /login, /logout, and /callback routes to the baseUrl
app.use(auth(config));

// req.isAuthenticated is provided from the auth router
app.get('/', (req, res) => {
  res.send(req.oidc.isAuthenticated() ? 'Logged in' : 'Logged out');
});
```

You can generate a suitable string for `secret` using `openssl rand -hex 32` on the command line.

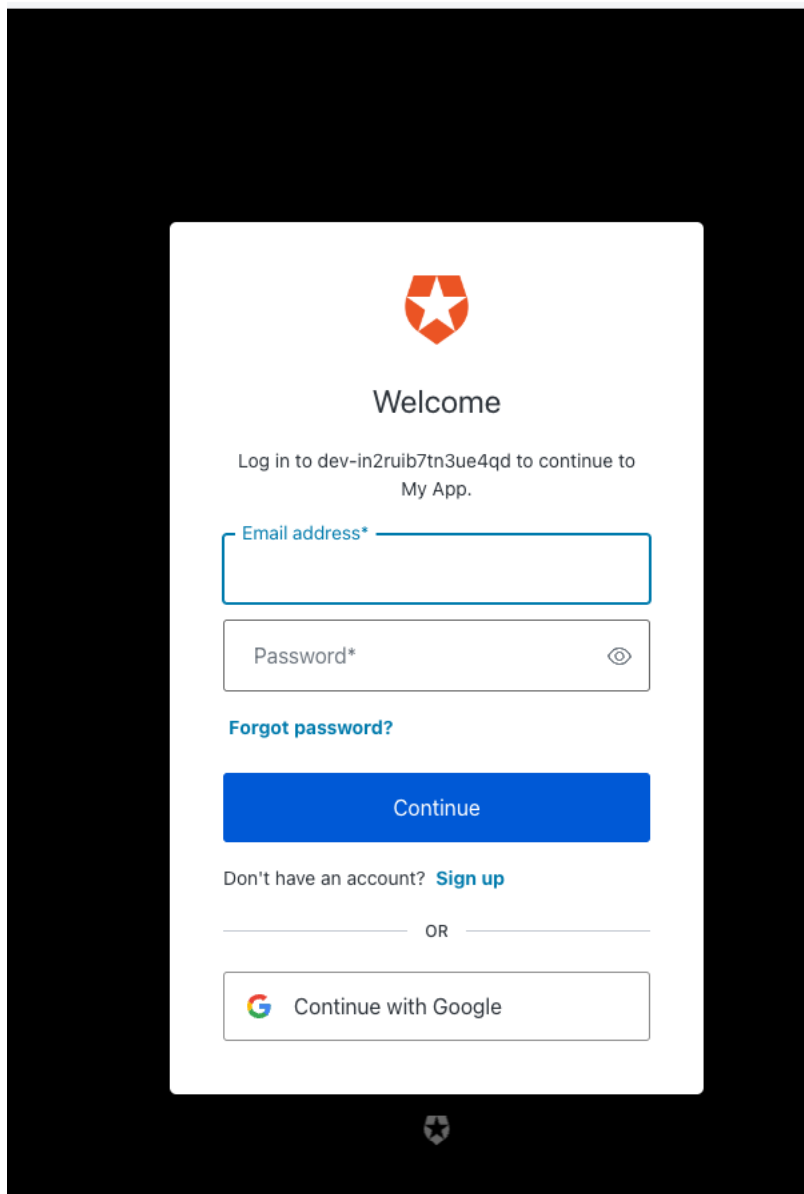
Next: Test Your Login

Nu hvor koden er sat op, så er det muligt at teste vores login. Dette kan gøres på <http://localhost:3000/login>

<http://localhost:3000/logout>

Login siden ser sådan ud:

utnu.com/u/login?state=nKf0Z5BukUg40HpxC3KJ5 1cyN0K7CjNINWZ1SvpoUHNmamJPakr



The image shows a login interface for an application. At the top, there is a URL bar with a long, alphanumeric string. Below it, a white login card is centered on a black background. The card features the Auth0 logo (a red shield with a white star) at the top. Below the logo, the word "Welcome" is displayed. A message reads: "Log in to dev-in2ruib7tn3ue4qd to continue to My App." There are two input fields: "Email address\*" and "Password\*", both with blue borders. The password field has a toggle icon (an eye) to its right. Below the password field is a link "Forgot password?". A blue "Continue" button is positioned below the links. Underneath the button, it says "Don't have an account? Sign up". A horizontal line with "OR" in the center separates this from the "Continue with Google" button, which includes the Google logo. At the bottom of the black background, there is a small, faint version of the Auth0 logo.

Her kan der både logges ind med en Google account ellers kan man sign up.

Jeg signede up, og inde på Auth0 hjemmeside har de en log-funktion hvor man kan se hvilke aktiviteter der er.

## Test your login

Now your application should be integrated with Auth0. The library provides default routes for logging in and out. You can test out the routes and we will be monitoring your logs in real-time to display relevant information.

- **For login:** In your application, visit the `/login` route provided by the library. If you are running your project on `http://localhost:3000` that link would be <http://localhost:3000/login> .
- **For logout:** In your application, visit the `/logout` route provided by the library. If you are running your project on `http://localhost:3000` that link would be <http://localhost:3000/logout> .

```
"hostname": "dev-in2ruib7tn3ue4qd.us.auth0.com",
"user_id": "auth0|663a4d0e505c4c1b0c8462df",
"user_name": "fuldpau@gmail.com",
"$event_schema": {
  "version": "1.0.0"
},
"log_id": "90020240507154728384325000000000000001223372048781043865",
"_id": "90020240507154728384325000000000000001223372048781043865",
"isMobile": false,
"id": "90020240507154728384325000000000000001223372048781043865",
"description": "Successful login"
}
```

 We detected a successful login!

Next: Get The User Profile

Nu hvor det er muligt at logge ind, så er det også muligt at display brugers profil og dens data. Denne kode indsætter jeg i den kode jeg allerede har:

[← Change technology](#)



Node.js (Express)

### Get the user profile information

To display the user's profile, your application should provide a protected route.

Add the `requiresAuth` middleware for routes that require authentication. Any route using this middleware will check for a valid user session and, if one does not exist, it will redirect the user to log in.

```
const { requiresAuth } = require( 'express-openid-connect' );

app.get( '/profile', requiresAuth(), (req, res) => {
  res.send( JSON.stringify( req.oidc.user ) );
});
```

Done



Koden kommer ikke til at virke fordi deres eksempel er med common.js. Derfor har jeg ændret lidt i koden ved at gøre dette:

```
import express from "express";
import "dotenv/config"; 7.6k (gzipped: 3.2k)
import pkg from "express-openid-connect";

const { auth, requiresAuth } = pkg;
```

Fejlbeskeden sagde at jeg skulle gøre dette. Her importerer jeg hele pakken og efter vælger jeg auth og requiresAuth fra pakken. Herefter kan vi bruge requiresAuth som en middleware, hvor der tjekkes for valid user session.

Dette er hele koden:



localhost:3000/profile

```
{
  "sid": "z-9mqQ0AXarWSyLxgPmM2eTf3JuXLjB",
  "nickname": "fuldpau",
  "name": "fuldpau@gmail.com",
  "picture": "https://s.gravatar.com/avatar/f444df6113da08f5ee45b78c91c994df?m=480&r=pg&d=https%3A%2F%2Fcdn.auth0.com%2Favatars%2Ffu.png",
  "updated_at": "2024-05-7T15:47:26.280Z",
  "email": "fuldpau@gmail.com",
  "email_verified": false,
  "sub": "auth0663a40e505c4c1b0c8462df"
}
```

Det her er hele koden:

```
package.json .env app.js
08c.Auth_integration > app.js > ...
1 import express from "express";
2 import "dotenv/config"; 7.6k (gzipped: 3.2k)
3 import pkg from "express-openid-connect";
4
5 const { auth, requiresAuth } = pkg;
6
7 const config = {
8   authRequired: false,
9   auth0Logout: true,
10  secret: process.env.SECRET,
11  baseURL: process.env.BASE_URL,
12  clientId: process.env.CLIENT_ID,
13  issuerBaseURL: process.env.ISSUER_BASE_URL,
14 };
15
16 const app = express();
17
18 app.use(auth(config));
19
20 app.get("/", (req, res) => {
21   res.send(req.oidc.isAuthenticated() ? "Logged in" : "Logged out");
22 });
23
24 app.get("/profile", requiresAuth(), (req, res) => {
25   res.send(JSON.stringify(req.oidc.user));
26 });
27
28 const PORT = process.env.PORT;
29
30 app.listen(PORT, () => {
31   console.log(`Server is running on port ${PORT}`);
32 });
33
```