LESSON 2

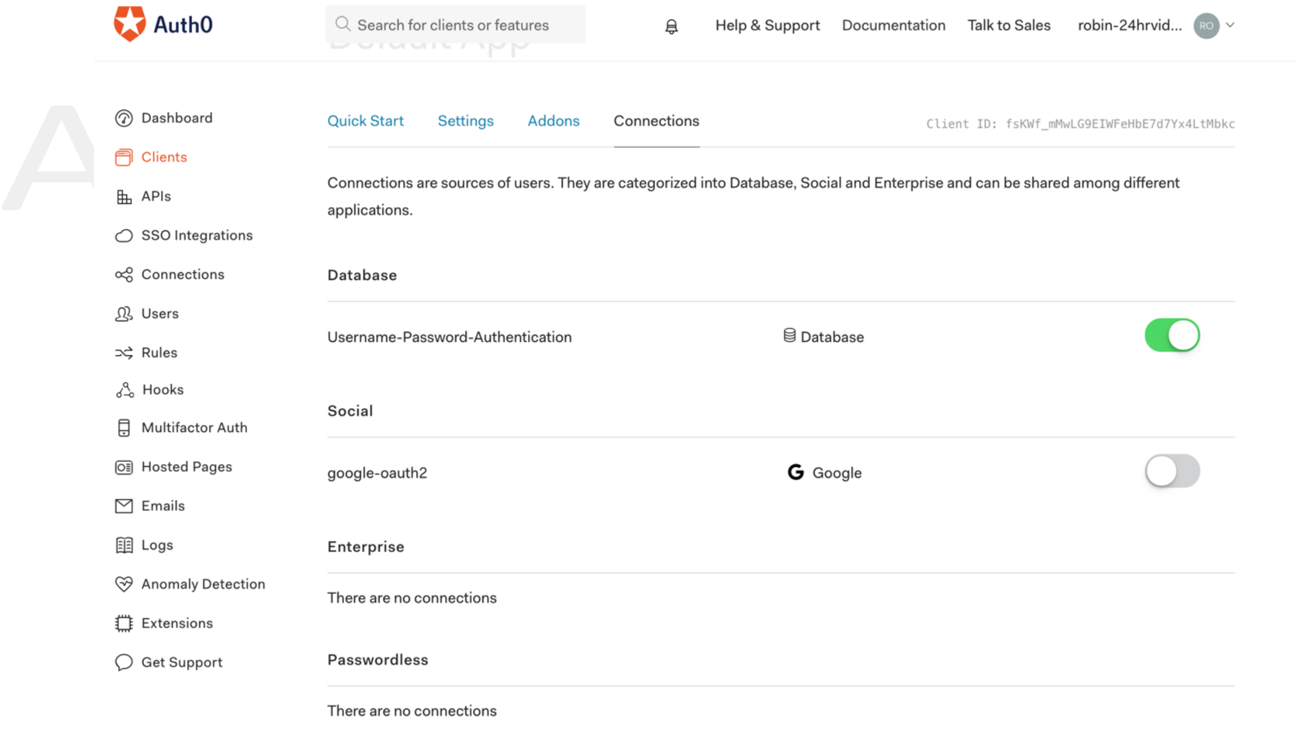
In this lesson, we'll create a website for our video hosting platform. We'll integrate the website with Auth0, as the means of authenticating users.

In true serverless style, this will be a static website, meaning that it can be hosted on S3, or any CDN. It is comprised entirely of static HTML, JS, and CSS and does not need to be served by a traditional web server.

1. CREATE AUTH0 ACCOUNT

You'll need to create a free auth0 account. Visit https://auth0.com and follow the sign up steps.

- You'll be asked to enter an **tenant domain**. Enter a name that is unique to you, e.g. janesmith-24hrvideo.auth0.com
- Enter **"US"** as your **region**.
- Click **Next** and fill out the information on the next page.
- Click **Create Account**.
- Go to **Clients** in the left navigation menu, and click on the **Default App**.
- Go to **Connections** in the **Default App** menu, and make sure that only **Username-Password-Authentication** is enabled.



Client ID: fsKWf_mMwLG9EIWFeHbE7d7Yx4LtMbkc

Connections are sources of users. They are categorized into Database, Social and Enterprise and can be shared among different applications.

Database

Username-Password-Authentication — Database — ●

Social

google-oauth2 — Google — ○

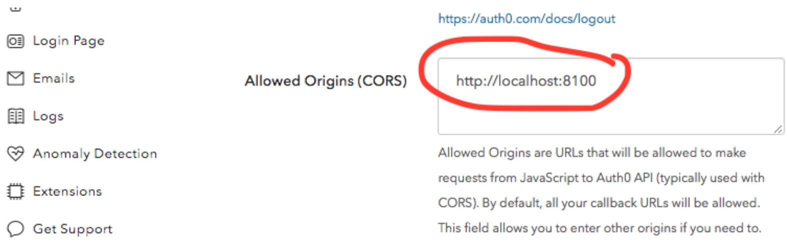
Enterprise

There are no connections

Passwordless

There are no connections

- Go to **Settings** in the **Default App** menu.
- Scroll down until you find the textbox called **Allowed Origins (CORS)**. Enter the following value in the textbox: **http://localhost:8100**

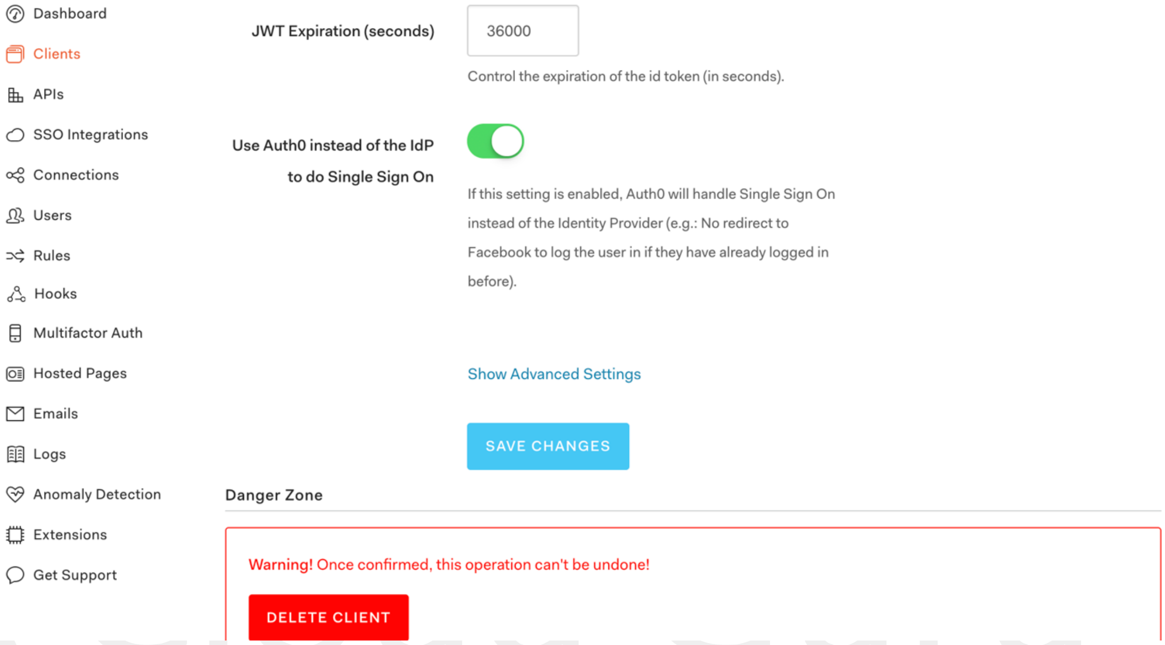


https://auth0.com/docs/logout

Allowed Origins (CORS): http://localhost:8100

Allowed Origins are URLs that will be allowed to make requests from JavaScript to Auth0 API (typically used with CORS). By default, all your callback URLs will be allowed. This field allows you to enter other origins if you need to.

- Scroll down to the bottom, and click the **Show Advanced Settings** link.



JWT Expiration (seconds): 36000

Control the expiration of the id token (in seconds).

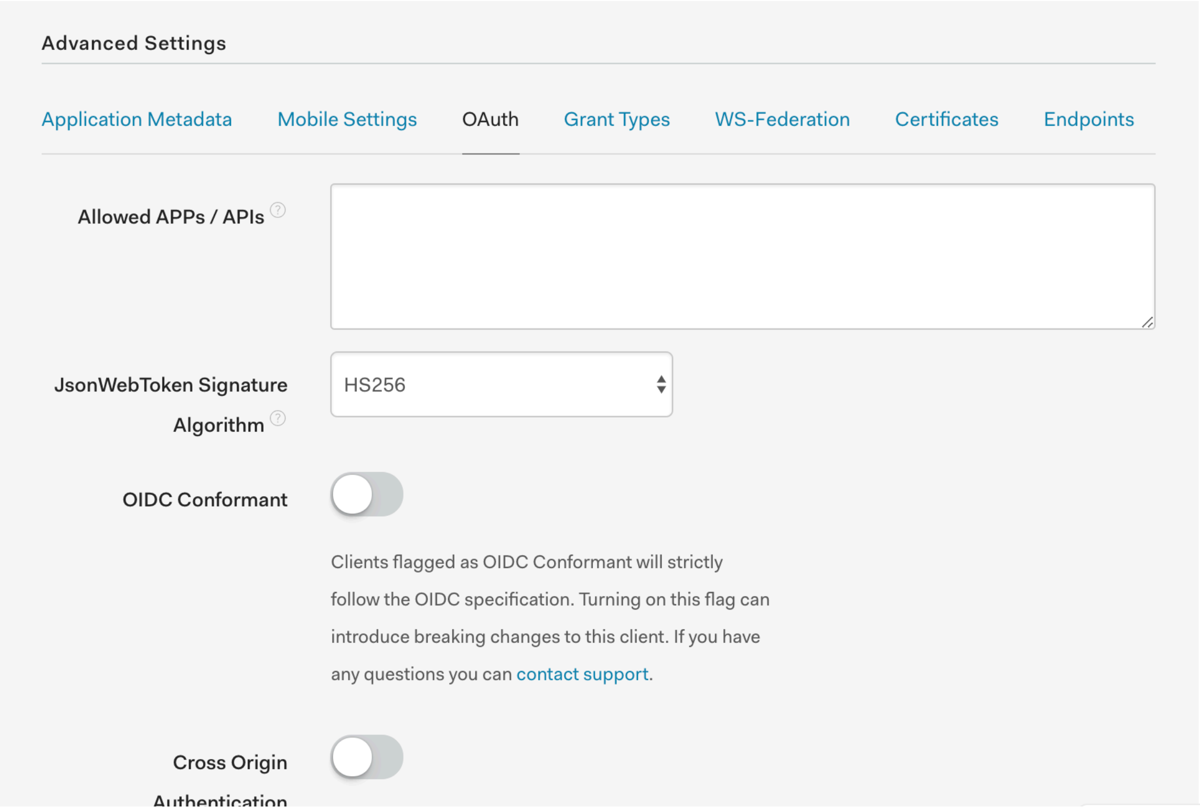
Use Auth0 instead of the IdP to do Single Sign On

If this setting is enabled, Auth0 will handle Single Sign On instead of the Identity Provider (e.g.: No redirect to Facebook to log the user in if they have already logged in before).

Danger Zone

Warning! Once confirmed, this operation can't be undone!

- Under **Advanced Settings**, choose the **OAuth** menu, and change **JsonWebToken Signature Algorithm** from **RS256** to **HS256**.
- In the same section, disable the **OIDC Conformant** option.

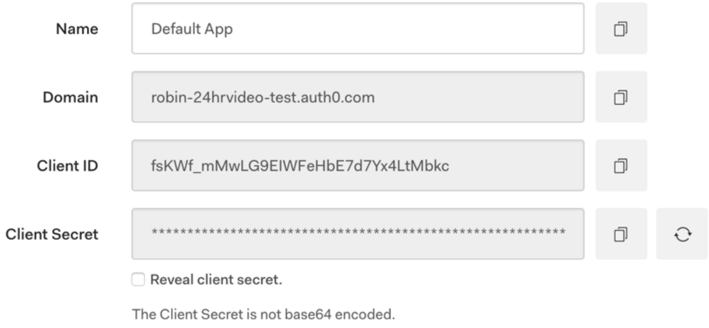


Advanced Settings

Allowed APPs / APIs	
JsonWebToken Signature Algorithm	HS256
OIDC Conformant	○
Cross Origin Authentication	○

Clients flagged as OIDC Conformant will strictly follow the OIDC specification. Turning on this flag can introduce breaking changes to this client. If you have any questions you can contact support.

- Scroll down and click the **Save Changes** button.
- We now need to retrieve some values from Auth0 that will be needed throughout this workshop. Scroll up to the top of the same **Settings** page, and find the **Domain**, **Client ID** and **Client Secret**. Copy these into your favourite text editor so you have them at the ready.



Name	Default App
Domain	robin-24hrvideo-test.auth0.com
Client ID	fsKWf_mMwLG9EIWFeHbE7d7Yx4LtMbkc
Client Secret	**

□ Reveal client secret.

The Client Secret is not base64 encoded.

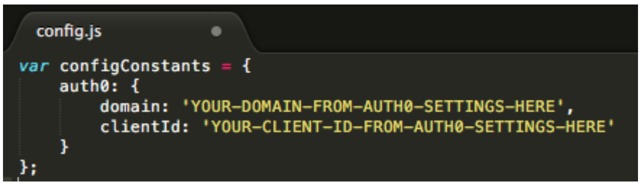
2. SETUP WEBSITE LOCALLY

This web site would normally be deployed via a CDN, but for the purposes of this workshop we're going to host it locally on your computer.

- Edit the following file in your favourite text editor:

lab-2/website/js/config.js

Enter your **auth0 domain** and **client ID**, in quotes (you made a note of these in the last step), and save the file.



```
var configConstants = {
    auth0: {
        domain: 'YOUR-DOMAIN-FROM-AUTH0-SETTINGS-HERE',
        clientId: 'YOUR-CLIENT-ID-FROM-AUTH0-SETTINGS-HERE'
    }
};
```

- Open a terminal / command-prompt and navigate to the following folder:

```
lab-2/website
```

- Run the following command, to bring down dependencies from npm (this may take a few minutes):

```
npm install
```

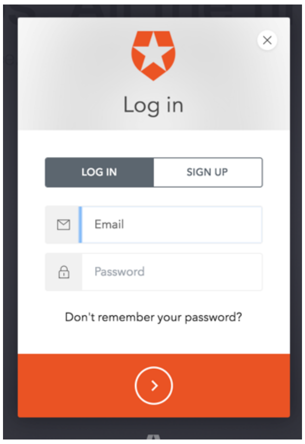
- Run the following command, to start a local web server at port 8100:

```
npm start
```

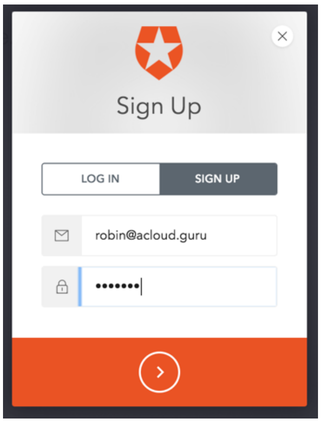
3. GIVE IT A SPIN!

- Open a web browser and navigate to:
  http://localhost:8100

  You should see the 24 hour video web site. There's not much here yet… that's OK! We're going to iteratively build the site during the workshop.
- Notice the **Sign In** button in the top-right? Click on it to launch the authentication popup:
  This popup is rendered entirely by Auth0 – a huge timesaver if you need authentication in your platform!
- You'll need to create an account, so click on the **Sign Up** tab:



- Enter an email address and password for your new account. This will be saved to your custom Auth0 database of users. **Remember this password**, because you'll need to sign in/out multiple times throughout this workshop



robin@acloud.guru

•••••••

- Click the big orange button at the bottom to create your account. A popup will launch to complete the sign up.
  Did you receive an error? Make sure you Always Allow Popups for this site.

- You'll be automatically signed in after your account is created. Look in the top right-hand corner of the web-site. You should see your name & a profile image / avatar (auth0 will use gravatar.com to find an image for your email address), plus a **Sign Out** button.
- When you're done with this lab, exit the "npm start" command in your terminal by pressing **<Control>-c**.

Congratulations – you now have a serverless web site with full user sign-up and authentication capabilities!

Get Your Hands Dirty

- Now use Auth0 to hookup a 3rd party social provider, such as Facebook or Twitter. Note: You will need to create an app with each provider that you hookup. Auth0's website has instructions.
- Auth0 supports running node.js rules on each user login. Add a rule to:
  - Force email verification (there is a pre-built Auth0 rule for this)
  - Only allow users from a specific white-list
- Auth0 supports the creation of delegation tokens to grant user's direct access inside your AWS account via IAM. It's worth understanding that this is possible and how it works. Read through the Auth0 documentation on this approach, and if you are game setup your web site to get an AWS delegation token.
  https://auth0.com/docs/integrations/aws