

## LESSON 2

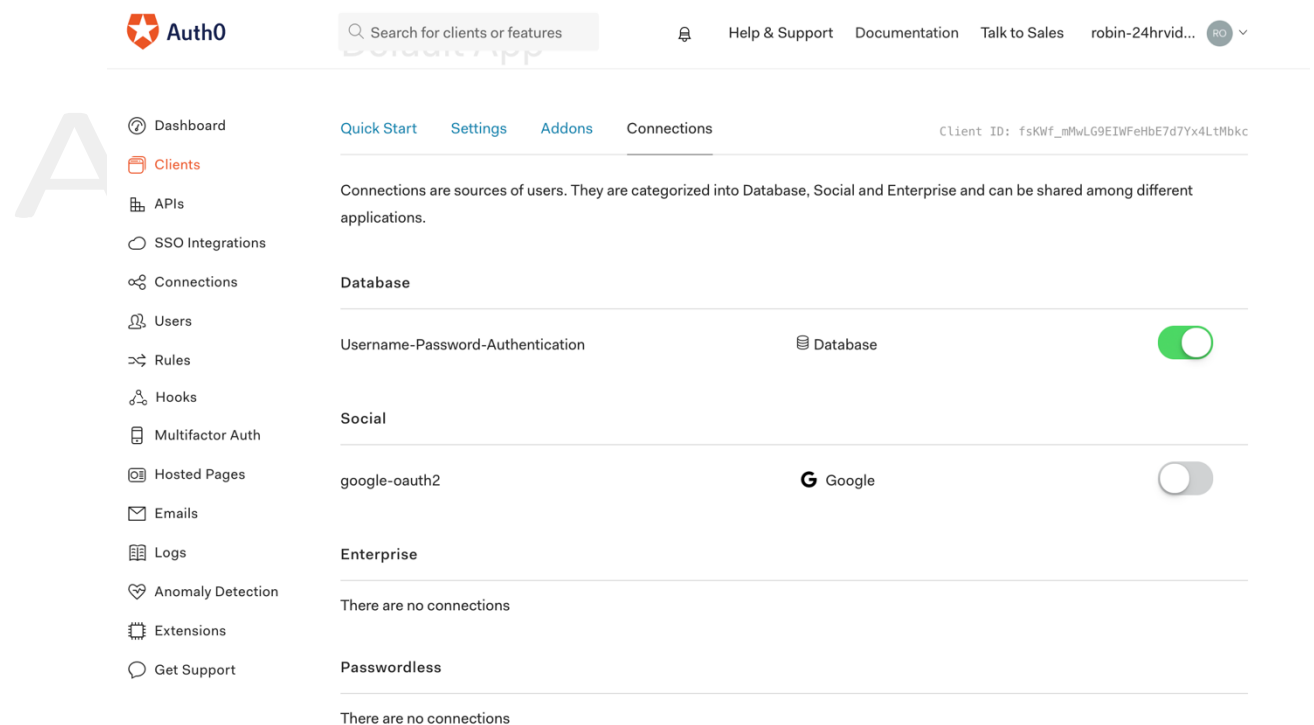
In this lesson, we'll create a website for our video hosting platform. We'll integrate the website with Auth0, as the means of authenticating users.

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 a **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.



The screenshot shows the Auth0 dashboard interface. The top navigation bar includes the Auth0 logo, a search bar, and links for Help & Support, Documentation, Talk to Sales, and a user profile. The left sidebar contains a navigation menu with options like Dashboard, Clients, APIs, SSO Integrations, Connections, Users, Rules, Hooks, Multifactor Auth, Hosted Pages, Emails, Logs, Anomaly Detection, Extensions, and Get Support. The main content area is titled 'Connections' and shows a list of connections categorized by Database, Social, and Enterprise. The 'Database' category is active, showing 'Username-Password-Authentication' with a toggle switch turned on. The 'Social' category shows 'google-oauth2' with a toggle switch turned off. The 'Enterprise' category shows 'There are no connections'.

Category	Connection Name	Provider	Status
Database	Username-Password-Authentication	Database	On
Social	google-oauth2	Google	Off
Enterprise	There are no connections		

- Go to **Settings** in the **Default App** menu.
- Scroll down until you find the textbox called **Allowed Callback URLs**. Enter the following value in the textbox: **http://localhost:8100,http://127.0.0.1:8100**
- Enter the same value into the following fields: **Allowed Web Origins**, **Allowed Origins (CORS)**

- Dashboard
- Clients**
- APIs
- SSO Integrations
- Connections
- Users
- Rules
- Hooks
- Multifactor Auth
- Hosted Pages
- Emails
- Logs
- Anomaly Detection
- Extensions
- Get Support

#### Allowed Callback URLs

http://localhost:8100,http://127.0.0.1:8100

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol, `http://` or `https://`, otherwise the callback may fail in some cases.

#### Allowed Web Origins

http://localhost:8100,http://127.0.0.1:8100

Comma-separated list of allowed origins for use with [Cross-Origin Authentication](#) and [web message response mode](#), in the form of `<scheme> "://" <host> [ ":" <port> ]`, such as `https://login.mydomain.com` or `http://localhost:3000`.

#### Allowed Logout URLs

A set of URLs that are valid to redirect to after logout from Auth0. After a user logs out from Auth0 you can redirect them with the ``returnTo`` query parameter. The URL that you use in ``returnTo`` must be listed here. You can specify multiple valid URLs by comma-separating them. You can use the star symbol as a wildcard for subdomains (`*.google.com`). Notice that querystrings and hash information are not taking into account when validating these URLs. Read more about this at <https://auth0.com/docs/logout>

#### Allowed Origins (CORS)

http://localhost:8100,http://127.0.0.1:8100

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. You can specify multiple valid URLs by comma-separating them or one by line, and also use wildcards at the subdomain level (e.g.: `https://*.contoso.com`). Notice that querystrings and hash information are not taking into account when validating these URLs.

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

Dashboard

Clients

APIs

SSO Integrations

Connections

Users

Rules

Hooks

Multifactor Auth

Hosted Pages

Emails

Logs

Anomaly Detection

Extensions

Get Support

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).

Show Advanced Settings

SAVE CHANGES

Danger Zone

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

DELETE CLIENT

- Under **Advanced Settings**, choose the **OAuth** menu, and ensure **JsonWebToken Signature Algorithm** is set to **RS256**.
- In the same section, make sure the **OIDC Conformant** option is green.

A Cloud Guru

### Advanced Settings

- Application Metadata
- Mobile Settings
- OAuth
- Grant Types
- WS-Federation
- Certificates
- Endpoints

Allowed APPs / APIs <sup>?</sup>

JsonWebToken Signature Algorithm <sup>?</sup>

RS256

OIDC Conformant

☒

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](#).






Cross-Origin Verification Fallback

https://domain.tld/path

Location URL for the page that will be rendered inside an iframe to perform the token verification when third party cookies are not enabled in the browser. Must be in the same domain where the embedded login form is hosted and must have a `https` scheme.

SAVE CHANGES

- 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_mMwLG9EIWFfeHbE7d7Yx4LtMbkc	
Client Secret	*****	 

☐ Reveal client secret.

The Client Secret is not base64 encoded.

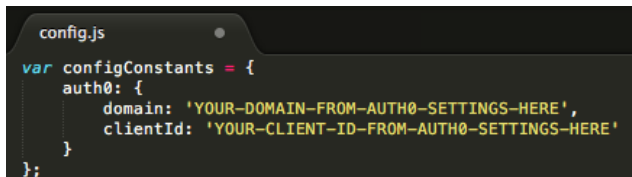
## 2. SETUP WEBSITE LOCALLY

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.



```
config.js
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`

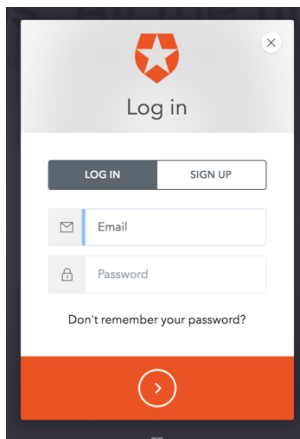
### 3. GIVE IT A SPIN!

- Open a web browser and navigate to:

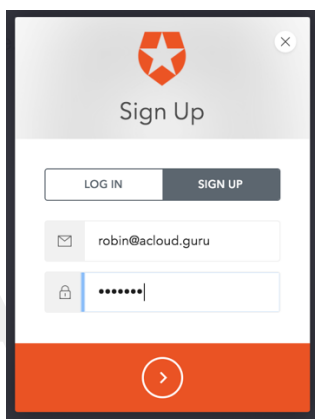
<http://localhost:8100>

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.

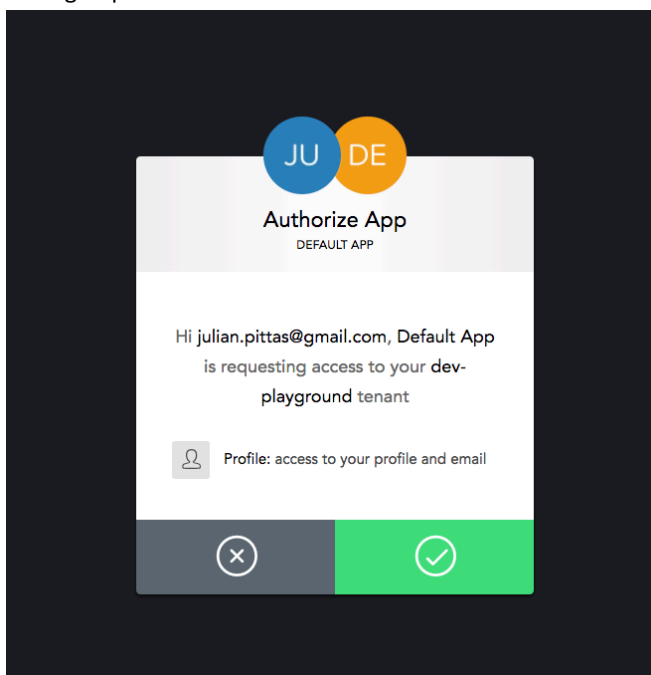
- Notice the **Sign In** button in the top-right? Click on it to launch the authentication popup: This popup is rendered entirely by Auth0 – a huge timesaver if you need authentication in your platform!
- You'll need to create an account, so click on the **Sign Up** tab:



- 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



- Click the big orange button at the bottom to create your account. A popup will launch to complete the sign up.



-

- Click on the **green tick**  
**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 3<sup>rd</sup> 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>