**Serverless Authentication with AWS Cognito and JavaScript**

In a traditional web application, authentication is handled by server-side code and users are managed in the database layer. In the world of serverless apps, we can offload the heavy-lifting to a managed authentication service like [AWS Cognito](https://aws.amazon.com/cognito/) to simplify it.

This post focuses on JavaScript code to authenticate users and manage sessions through AWS Cognito.

The entire solution can be found in this [repo](https://github.com/mydatahack/serverless-webapp-s3uploader).

The most important concept with AWS Cognito is to understand the difference between User Pools and Identity Pools. In a nutshell, User Pools manage user authentication and Identity Pools manage user authorization through IAM roles and permissions. For authentication, user pool is all you need. However, without setting up IAM role through Identity Pool, the authenticated user cannot take action on AWS resources, such as uploading files to S3 or interacting with DynamoDB. [This NewStack blog post](https://thenewstack.io/understanding-aws-cognito-user-and-identity-pools-for-serverless-apps/) is excellent in explaining the difference.

**Setting up users in Cognito**

Users can be set up in Cognito through user interface (see [AWS official documentation](https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-getting-started.html)). Usually, we need to write both sign in and sign up logics to manage users. When the user base is limited, sign up logic might not be needed. Without sign up logic, we need to run AWS command to confirm the user by manually changing their password.

Once the user is created in Cognito, we can run the command below to reset the password so that the user status becomes confirmed.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

**Dependencies**

Interacting with any AWS service is easy thanks to their excellent SDK. We are using Congito specific JavaScript SDK, [amazon-cognito-identity](https://github.com/amazon-archives/amazon-cognito-identity-js), which is part of [aws-amplify](https://github.com/aws-amplify/amplify-js). All you need to do is to import them in the script tag at the bottom of the body. [Here](https://github.com/mydatahack/serverless-webapp-s3uploader/blob/master/site/signin.html) is the example of sign in html page.

**Signing in**

The function can be invoked on clicking the sign in button. It makes an API call to Cognito and returns a token. If successful, redirect the user to the protected page. You need to obtain user\_pool\_id and client\_id from Cognito console, which should be straight-forward. The code is base on the [example from AWS](https://docs.aws.amazon.com/cognito/latest/developerguide/authentication.html).

<pre class="theme:vs2012-black font:verdana font-size:14px toolbar:2 lang:js decode:true ">

</pre>

**Checking user**

We can call getCurrentUser() method to see if the user is logged in or not. If there is not current cognito user in the session, it redirects to the sign in page.

<pre class="theme:vs2012-black font:verdana font-size:14px toolbar:2 lang:js decode:true ">

</pre>

**Signing out**

To sign out, we call signOut() method on the user. It will sign out the user and redirect to sign in page.

<pre class="theme:vs2012-black font:verdana font-size:14px toolbar:2 lang:js decode:true ">

</pre>