**The Beauty of Going Serverless**

Intro

I recently built a *traditional* (i.e. server-based)web app, for a project-centered college course. During lock-down, as I got better acquainted with more AWS services to prepare for the AWS CDA exam, I thought it’d be interesting to build such an app on the cloud, and even more interesting to build such an app with a serverless architecture.

Amazon had a sample [web app with a serverless backend](https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/) project, but the scenario for the app was, well... a little *less than practical* (it’s an online unicorn-ride-requesting service), and while this *fantastical* project *works*, it’s more of a recipe than a guide, lacking detailed explanations, and it doesn’t really demonstrate the full potential of a serverless web app.

So I’ve chosen to do that with the help of adifferentexample: an online store, where users can browse a catalog and save items to their cart, make purchases and see their previous purchases, as well as get notified when they make a new purchase. In other terms, a web app with a persistent **storage**,that can **manage sessions**, **authenticate** users and **trigger** specific functions based on specific backend events, without a single server instance.

Below I’ll go over the details of the application, the steps taken to implement it, and some explanations/comments on the services used, the code and the architecture itself. Speaking of which...

The Architecture

Getting Started

Note that all code mentioned here is also available on Github, so feel free to check it out on <link>. Before anything else though, make sure you:

* Have an AWS account. This project shouldn’t cost more than 0.25-0.3 $, even if you’ve exceeded the limits of your free-tier account, despite almost every service used here being eligible for free-tier use.
* Are familiar (at least on a basic level) with HTML and JavaScript

Now, I understand that while some AWS services have pretty straightforward names (e.g. *Amazon RDS* – Relational Database Service), some simply give you no clue as to what their purpose is (e.g. *Amazon Athena* – I’m all for Greek mythology name-dropping, but how on earth am I supposed to know that this is a service for querying buckets of data with SQL?). So, first things first:

The Services Used

* Simple Storage Service, AKA **S3**

As the name suggests, this is a storage service, specifically, it’s an object (i.e. file) storage service. Objects are uploaded to specific **buckets** (which can contain sub-folders). One cool feature of S3 that we’re going to make use of, is how we can upload a simple HTML+CSS+JS **static** website to a bucket, and S3 can host it for us.

* **Lambda Functions**