Title: Web Scraper for a Property Website

Prepared By: Amal Byju

Date: 3rd December 2023

Introduction:

The objective of this mini project is to develop a web scraper via Beautiful Soup and Selenium that could extract apartment listings from a property website for certain cities every week and store them in BOX or Amazon S3 for analysis and gathering insights. The collected data would enable the client to view trends in demand and apartment prices for every city.

Project Description:

The application should be able to extract information related to property listings from the property website for the following locations of interest:

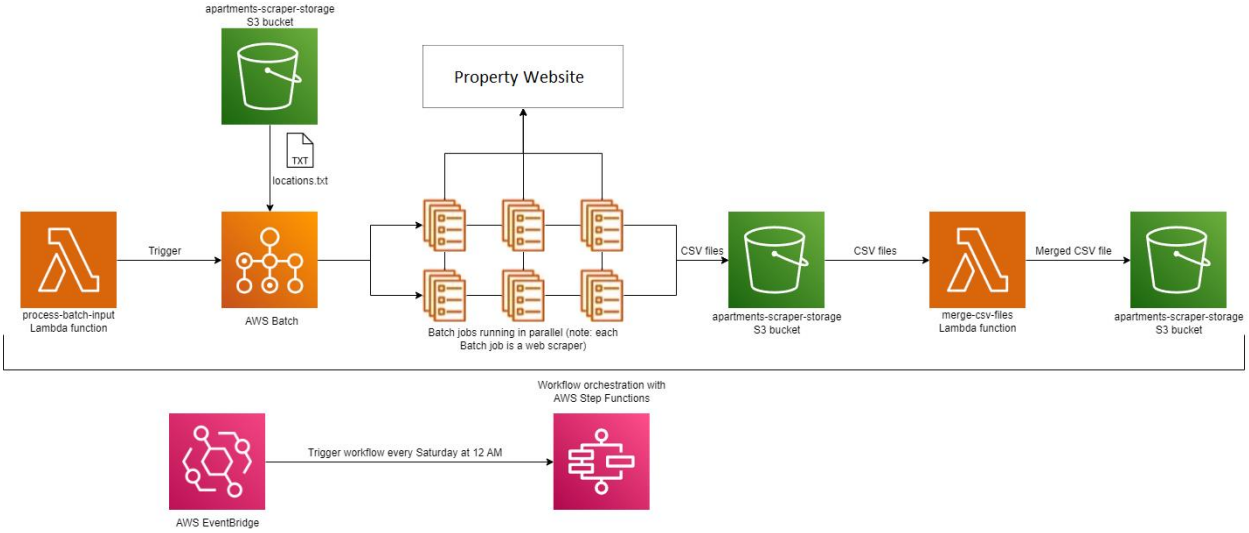
1. CA: Rosemead, San Gabriel, El Monte, South El Monte, Arcadia, Monterey Park, Claremont, Alhambra, Glendora, Covina, West Covina, Azusa, Montebello, Pasadena, Glendale, Riverside, Ontario, Upland, Fontana, Redlands, Corona, Rancho Cucamonga, Montclair
2. GA: Warner Robins
3. WA: Lake Stevens, Everett

The following fields are extracted from the website for each property listing:

1. Property Name
2. Street
3. City
4. State
5. Postal Code
6. Neighborhood Address
7. Monthly Rent
8. Number of Bedrooms
9. Number of Bathrooms
10. Square Feet
11. Amenities
12. Total Units
13. Contact Number

The data is collected and stored in a tabular format so that it can be easily accessed by another system for wrangling and analysis later.

Architecture:



The process-batch-input Lambda function kicks off the workflow by triggering AWS Batch, which is a service that could create a Fargate cluster that accomplishes a certain task with ease – in our case, it is to scrape a property website for property listings. A locations.txt file resides in apartments-scraper-storage S3 bucket that has the list of all locations (the list is mentioned in page 1). Once AWS Batch is ready, it creates a cluster having six instances and then each instance in the cluster reads the locations.txt file. The locations would be divided among the instances (this enables parallel processing which in turn drastically reduces the execution time). A particular instance would scrape all listings for the locations it is assigned and loads a CSV file containing the tabular information in apartments-scraper-storage S3 bucket. So, after the Batch job is complete, we would have 6 CSV files named like so: 0.csv, 1.csv, 2.csv, 3.csv, 4.csv and 5.csv. Then, merge-csv-files function would merge these CSV files into a single file with the current date as the file name. These steps are part of a workflow in AWS Step Functions service to ensure that they are executed in order. The workflow would be triggered every Saturday at midnight to ensure a steady flow of information related to property listings in the property website.

Note:

1. Locations could be added or deleted from locations.txt file to adjust the web scraping activity accordingly.
2. For the web scraping activity, a certain python script is run in every instance in the Fargate cluster.
3. The scraper retrieves listings for apartments only; other property types are excluded.
4. The number of instances in the Fargate cluster is determined by the number of lines in the locations.txt file. Presently, there are six lines in the file and editing the file might impact the number of instances in the Fargate cluster.
5. A simple UI page from where the user can trigger the application and an error logging and handling mechanism for developers to easily find and debug errors are additional functionalities that could be implemented in the future.