Team: Thinh Dang, Ashlyn Bui, Adrian Unruh, Isabella Abad

Program 5 Movie Watchlist

Professor Dimpsey

Public Movie WatchList Web Application

Program Objective:

A restful web application helps the user to find their favorite movie/anime and add it into the watchlist.

Web location

Static web application: http://program5vueclient.s3-website-us-west-2.amazonaws.com
API Backend: http://program5-env-1.eba-mihuv8he.us-west-2.elasticbeanstalk.com/index

Services Utilized

1. Elastic Beanstalk
2. DynamoDB
3. S3
4. Movie API
5. Anime API
6. Exposing API for CRUD operations

Design

As shown in *Figure 1*, the backend application was deployed using AWS Elastic Beanstalk on the AWS cloud, uses DynamoDB to store movies/anime that have been added to the watchlist. Two APIs are consumed. One accesses a movies database service that returns movies based on the title. Each return object contains the movie title, director, actors, synopsis, etc. The Anime API does the same thing but for Animes instead of movies.

A static web application built using Vue.js is hosted on AWS S3 and consumes the APIs provided by the backend application. This includes APIs for CRUD operations on Animes and Movies, and two APIs for displaying Movie and Anime search results to users.

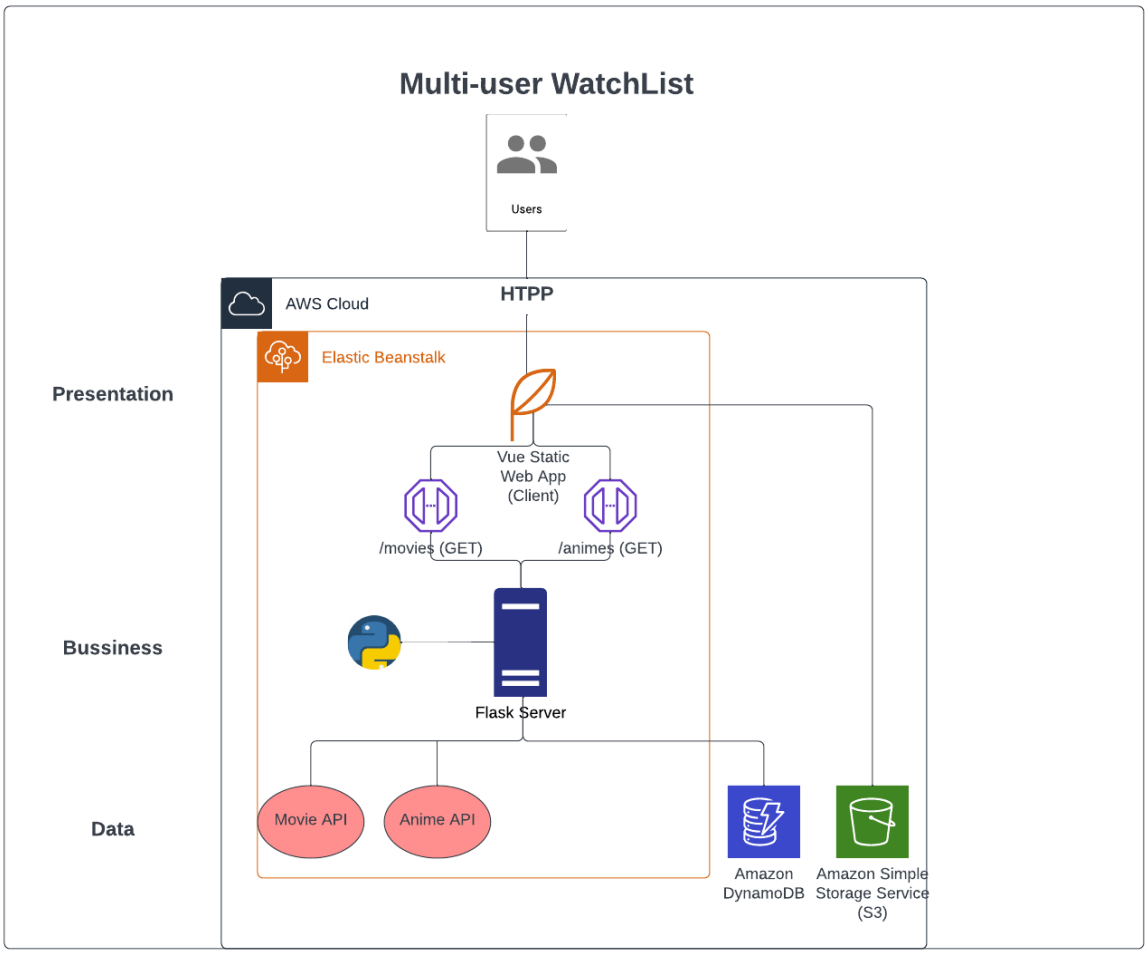

Figure 1. WatchList Design

APIs:

- Base url: http://prog5-env.eba-qrf9haiy.us-west-2.elasticbeanstalk.com
- **/api/movies/searchMovie**: Calls two OMDb APIs to get movie titles, year, director, actors, and plot
    - /api/movies/searchMovie?title={movieTitle}

- **/api/anime/searchAnime**: Calls an AniList API to get anime titles, year, director, actors, and plot
    - /api/anime/searchAnime?title={animeTitle}
- **/add**: Adds a movie
- **/delete**: delete a specific movie (composite key: title, year). Note: need to use a postman to pass in the object.
- **/deleteAllItems():** reset the entire list (delete all the items in the dynamodb)
- **/query/movies:** get all items in the database
- **/query/movies?title=the%20platform&year=2019**: querying a specific movie
- **/update**: incrementing the vote count (showing favorite movie)

Usability

The restful application is exposed to users by Vue (user interface). Via http call, clients search for movies or animes that they want. The composite key for searching a movie/anime is title and year. A selected movie will be added to the watchlist (dynamodb database) for the future review.

Cloud Service

The main reason why our group chose to use AWS was because we believed that the UI was easier to use and understand compared to the Azure UI. All members had previous experience with using AWS services, and with the limited time given for this project, we decided to choose the one most were familiar with. Additionally, AWS allows us to pick what coding language we want to use.

Monitoring

Monitoring for availability is done through AWS Elastic Beanstalk and DynamoDB monitoring functions on the console. As partially seen within *Figure 2* and *Figure 3*, the measured metrics for Elastic Beanstalk include: environment health, target response time, the sum requests, cpu utilization,max network in/out, and many other kinds of metrics seen as numbers and graphs within the service monitoring interface. The time range for the metric data can be adjusted between 1 hour to 2 weeks and the period can be adjusted between the range of 1 minute to 1 hour. As partially seen within *Figure 4*, the metrics measured for DynamoDB include: read/write usage, read/write throttled requests, read/write throttled events, latency, and many others. All metrics are shown as graphs, and the metric timeframe range can be changed from 1 hour to 1 week.

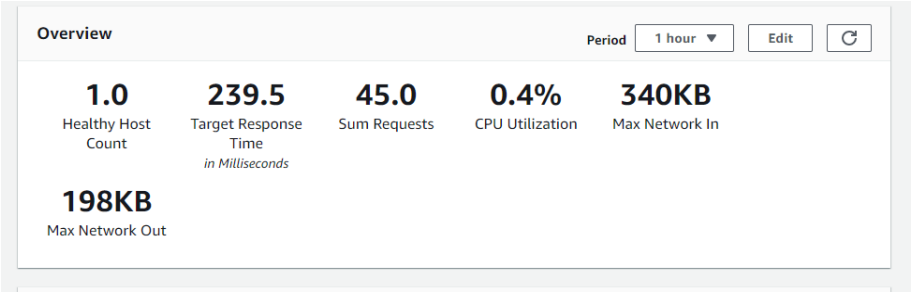

Figure 2. Elastic Beanstalk overview metrics

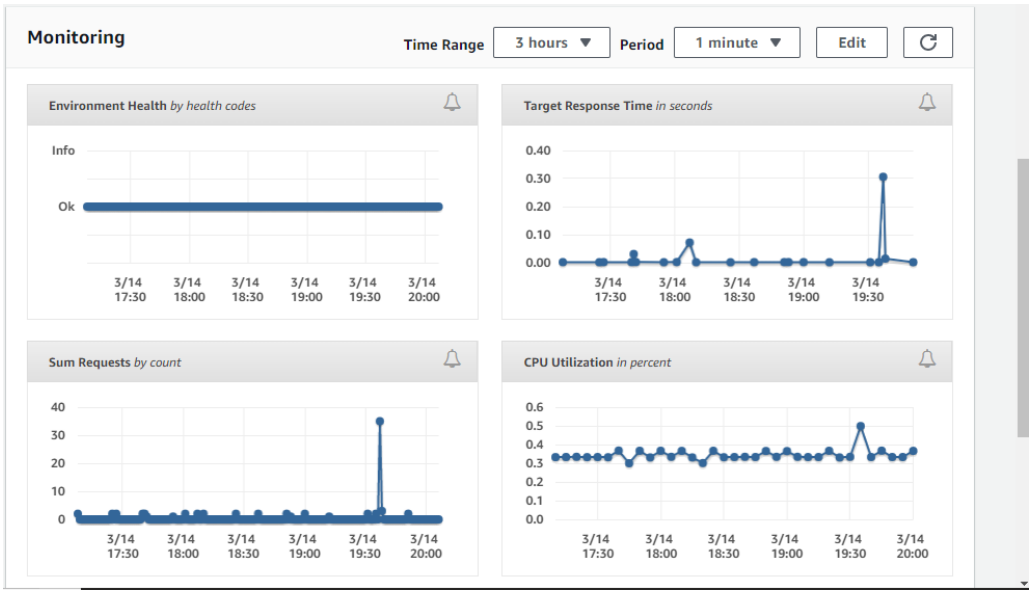

Figure 3. Elastic Beanstalk metric monitoring

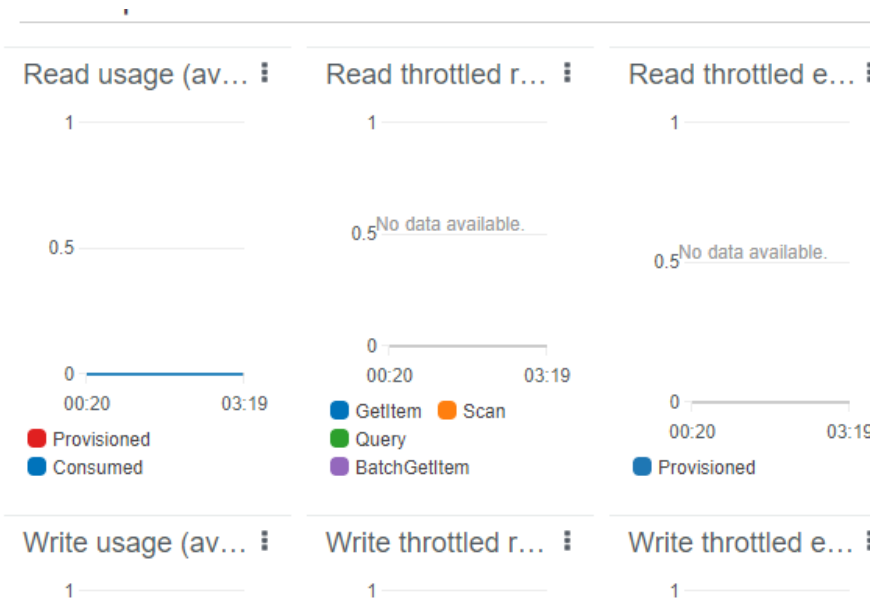

Figure 4. DynamoDB metric monitoring

There are currently no alerts set up for our program. Alerts, or called alarms by AWS, can be set up easily within the monitoring interface for both Elastic Beanstalk and DynamoDB for a certain metric. If we were to have an alarm set up, we would be able to set a threshold for the metric, the time interval between readings, and the amount of

time after a threshold has been reached that an alert should be sent. Alarms are sent through a given email address.

SLA:

SLA for S3 = 99.99 (S3 Standard storage class 99.99%)

SLA for DynamoDB = 99.99 (at least 99.99% is standard SLA)

SLA for EC2 = 99.99 (for EC2 a single instance is 90% but number of instances range from 1-4 in any availability zone so 99.99%)

Api calls fail 1-2% of time/day (estimation)

1/.02 = 50 days for MTTF with 30 minute downtime

.9999*.9999*.9999*(1-(30m/(50d*24h*60m))*(1-(30m/(50d*24h*60m)) = .9988

Overall:

- **Availability: 99.88%**
- **Durability: 1(dynamodb) * 0.9999 (EC2) * 0.9999(S3) = 0.9998 = 99.98%**

Scalability

This application uses horizontal scaling and uses a load balanced environment which increases or decreases the number of EC2 instances within a range of 1-4 based on the current load. It is scaled by one instance that has a lower threshold of 2000000 bytes and an upper threshold of 6000000 bytes. Measurements are taken every 5 minutes and the application is auto-scaled accordingly within 5 minutes of breaching the NetworkOut metric threshold. DynamoDB has auto scaling and can be adjusted from on-demand to provisioned. Creating an auto scaling policy allows the setting of lower and upper thresholds for targeted autoscaling of throughput capacity.

There will be a certain amount of clients that can load the program at once due to the free -tier plan service of AWS. It also impacts to other's client applications speed if there are many clients perform CRUD operations

Reference

https://lucid.app/lucidchart/52c1f0e1-c672-4de5-a669-a2e359cccca7/edit?invitationId=inv_5002df6a-303f-42b7-bfc0-b965c88b2d44.

https://aws.amazon.com/dynamodb/sla/