Implementing WildRydes

Using AWS services

AWS services in use

- S3 (hosting the static website, which later becomes dynamic).
- Cognito (creating a user pool).
- DynamoDB (records unicorn requests).
- Lambda (back-end coding for unicorn requests and interacts with front-end app).
- IAM (for role permissions).
- API Gateway (RESTful API accessible in the public internet).

Overview

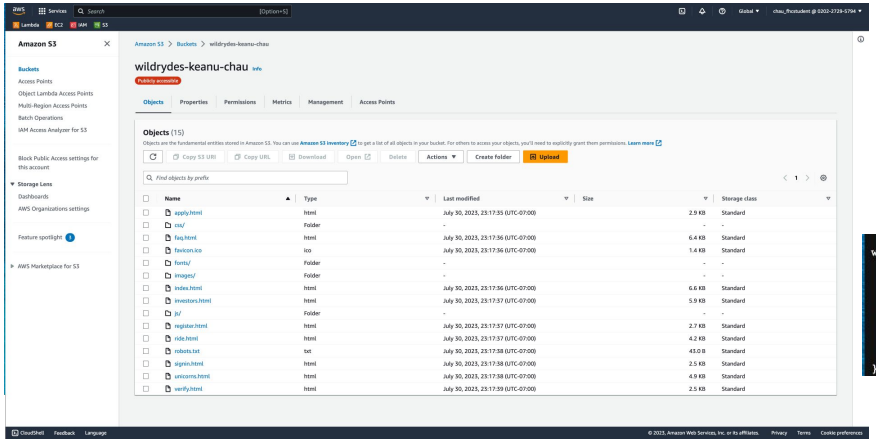
Our S3 storage will store all our front-end files that users interact with and host a static website that we will later configure to a dynamic one. After that, we create a user pool for customers using Cognito that we'll use for our website. We also use IAM to create a role that grant access for Lambda and DynamoDB. When the user logs in, it creates and authentication token that can be authenticated with API Gateway. Then Lambda and DynamoDB handles our requests by Lambda providing the back-end code for an API request and DynamoDB recording the request for a unicorn. This is all done easily with AWS services, without the need of costly physical hardware.

S3 Storage

From the talented developing team we got Javascript, HTML, CSS, image, and other files to be able to easily host the static website in S3. By this time, all the front-end has been developed.

We create a root domain bucket to store our files, make it publicly accessible, and turn it into a static website. Users will have a public URL that can be modified later using Route 53.

We later edit a ***config js.*** file using Cognito and API Gateway information that is specific to our user pool and API.



S3 bucket

config.js file edited to contain our user pool and API data

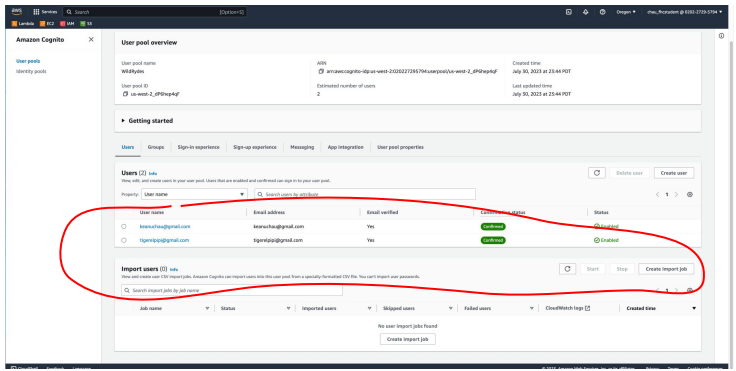
```
window._config = {
    cognito: {
        userPoolId: 'us-west-2_dP6hep4qF', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '2coq6gqe6qe0pvof8ucvs59j4p', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: 'us-west-2' // e.g. us-west-2
    },
    api: {
        invokeUrl: 'https://zvrwpngl5g.execute-api.us-west-2.amazonaws.com/prod' // e.g. https://ttf5alun5d.execute-api.us-west-2.amazonaws.com/prod
    }
};
```

Amazon Cognito and IAM

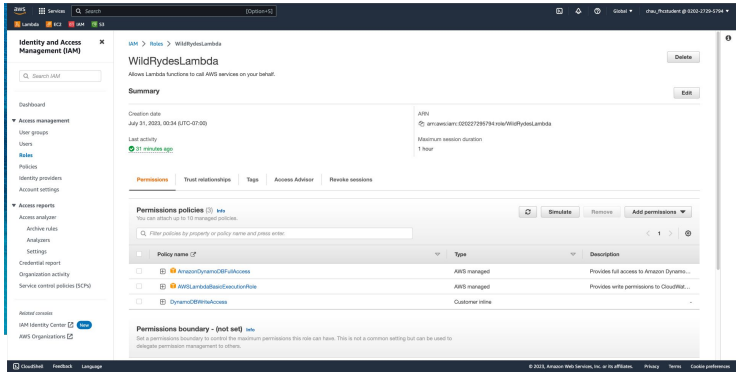
We create a user pool to manage our user's accounts with Cognito. We put this information on our ***config js.*** file in S3. Usual user information and password is provided and then user get sent a verification message. MFA is recommended. When logging in WildRydes, a Javascript function from S3 talks with Cognito and an authorization token is provided and can be tested using API Gateway.

AWS mainly just charges for monthly active users.

We also use IAM to create a role that allows for Lambda and DynamoDB access.



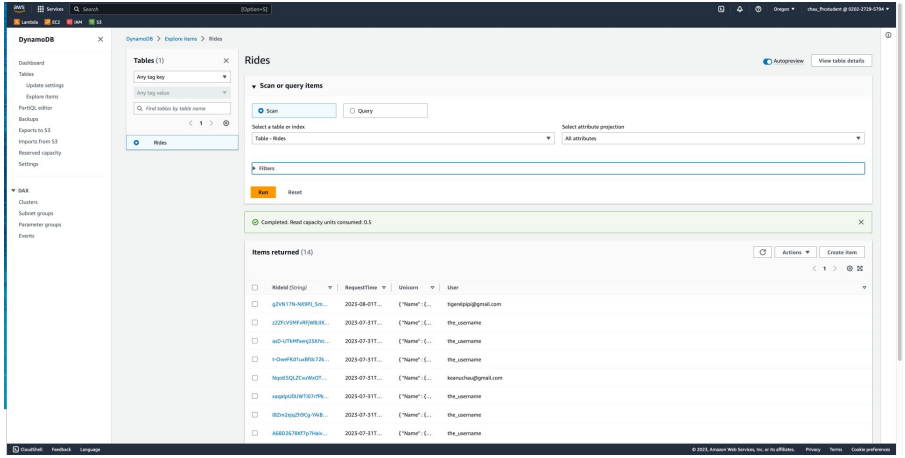
Cognito showing user information



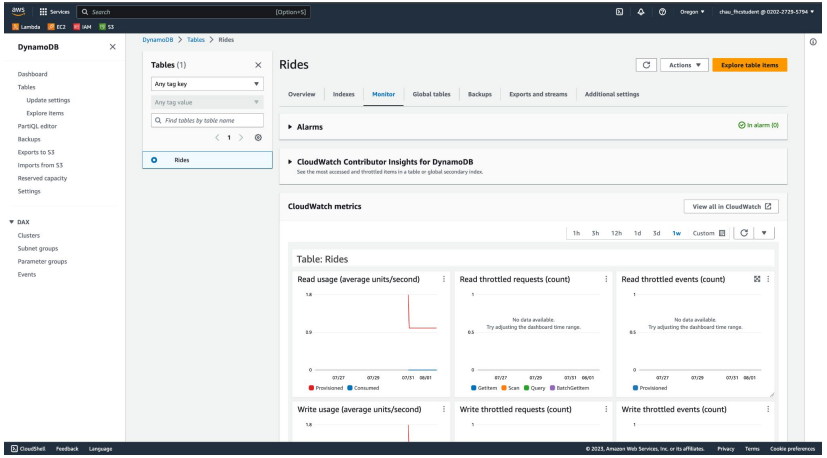
IAM WildRydes role

DynamoDB

We create a table using DynamoDB in order to record unicorn requests. Note that the IAM role includes recording requests in this table. When the user logs in and requests a unicorn, the Lambda function will record the request in the table.



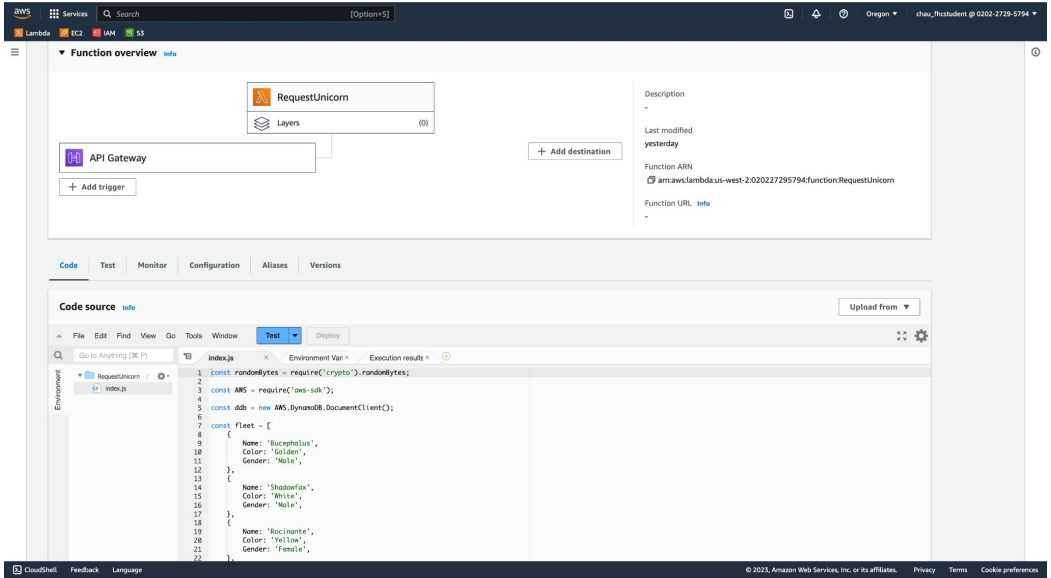
DynamoDB table showing items from requests



DynamoDB showing CloudWatch metric of the table

Lambda

The function runs in response to HTTP requests (API requests) from the user logging in and requesting a unicorn. It will connect with a RESTful API in API Gateway in order to process the request, send a unicorn, and record it in DynamoDB. It uses the IAM role we created earlier and uses Node.js 16.x for the Runtime.

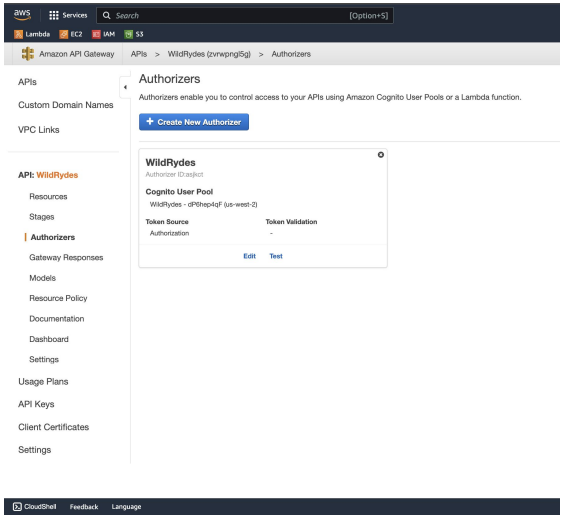


Lambda function showing it is connected with API Gateway

API Gateway

A RESTful API is needed to connect with the Lambda function and is exposed to the public internet user requests. A page we already put out in S3 interacts with the RESTful API by jQuery's ajax() method to make the remote request which then calls the Lambda function which sends the unicorn per the users request.

It uses Cognito to authorize the user that is calling the API using authentication tokens.

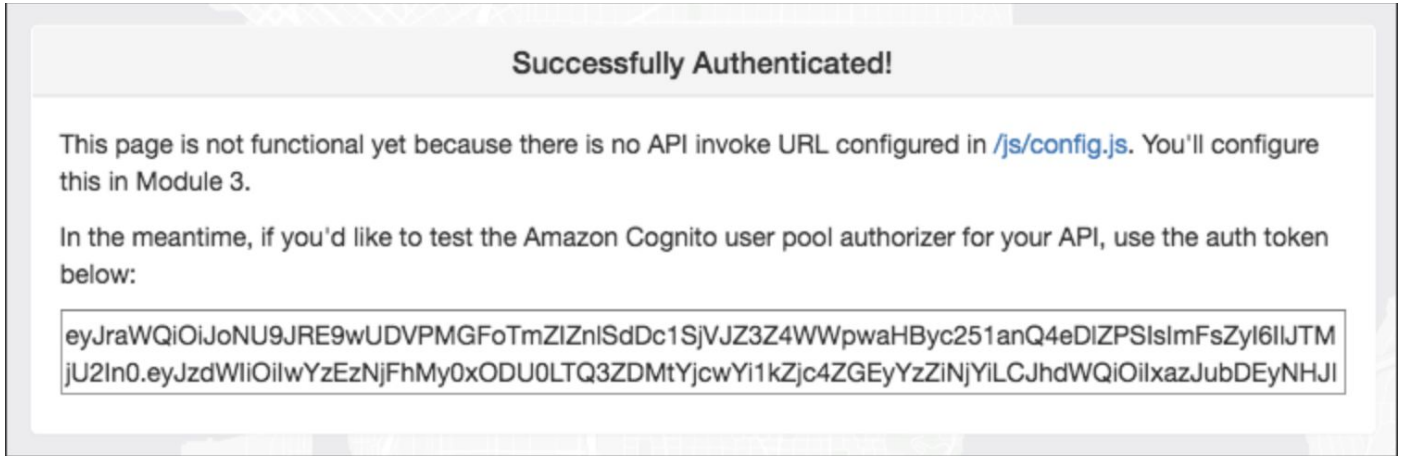


API Gateway showing that the user pool created in Cognito is being used as an authorizer.

Testing at every step

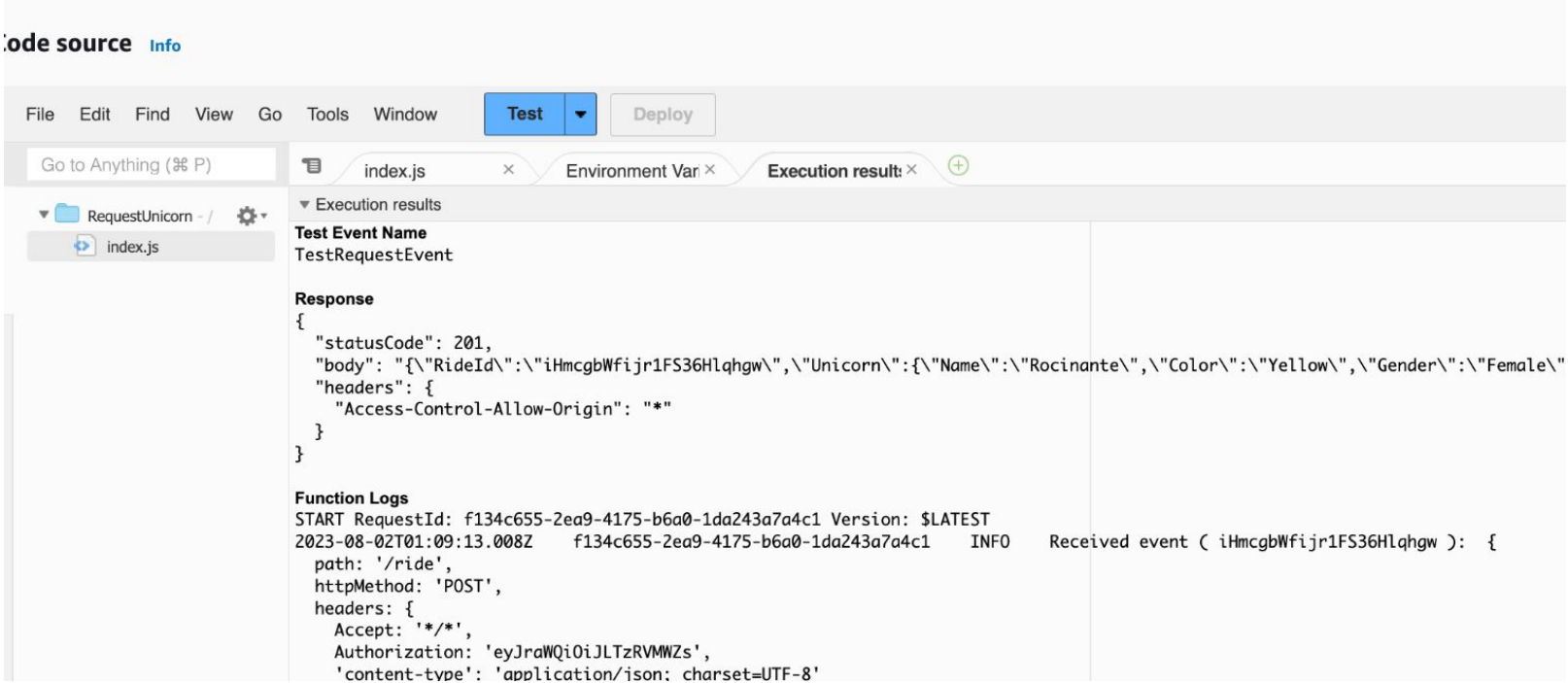
S3: After uploading all the files from the developing team we tested the now static website by using the provided URL to check if it was active.

Cognito: We logged in as a user to test if we got a verification code and see if we could register and then sign in. Verification message was successful.

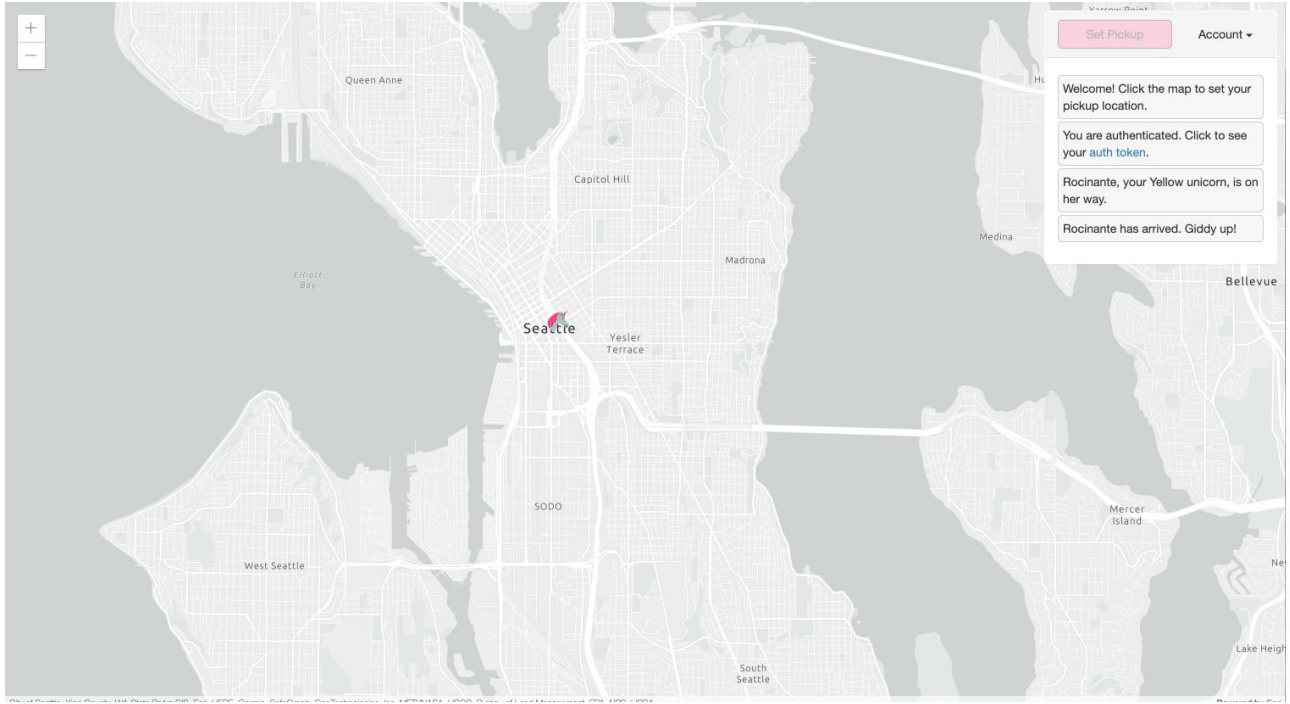


This should appear after because we have not created our RESTful API yet.

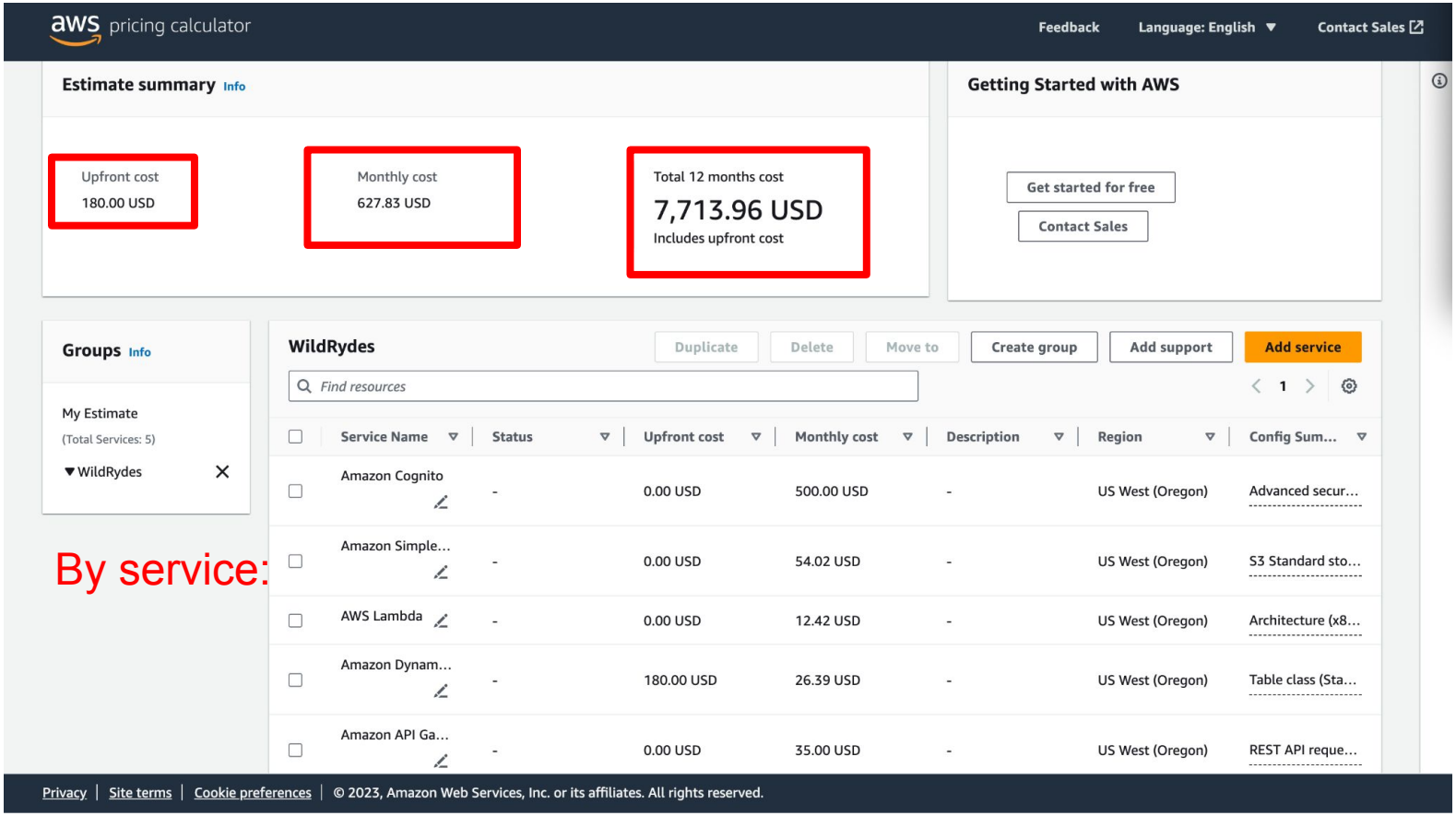
Lambda: We configured a test event using Lambda then tested the code we got from our developing team and got correct execution results.



API Gateway: With the API ready, testing the whole website isready since we completed all necessary steps. We request a ride from a location so that it calls the API which the runs the Lambda function and sends the unicorn. It also then records the request in DynamoDB. Website runs successfully.



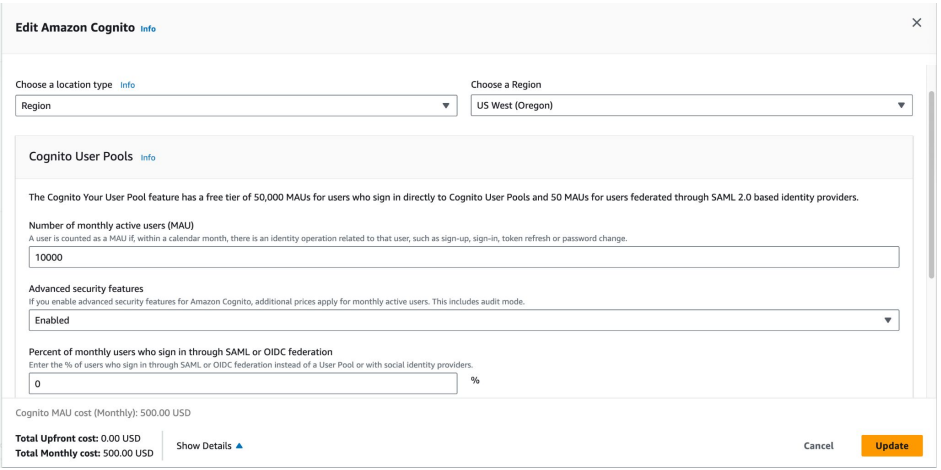
Cost Analysis Overview



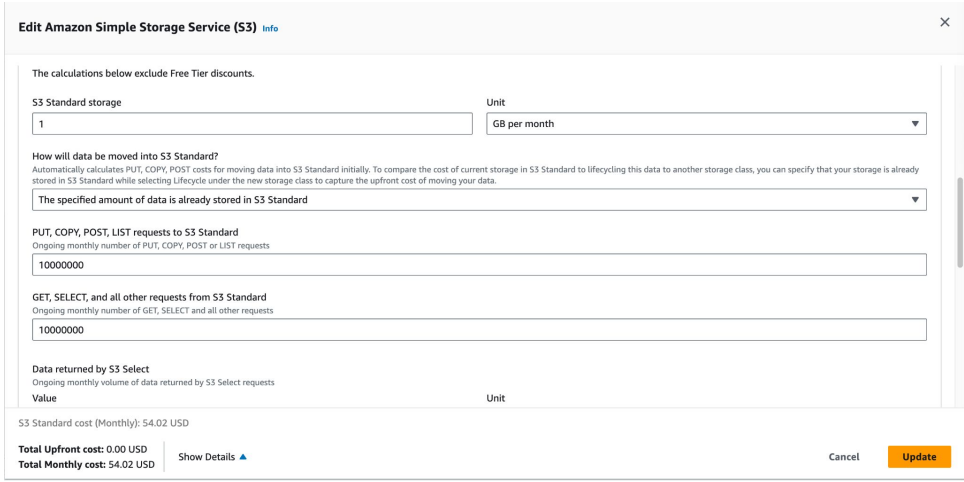
By service:

Cost Analysis by service

Cognito: 10,000 user estimate



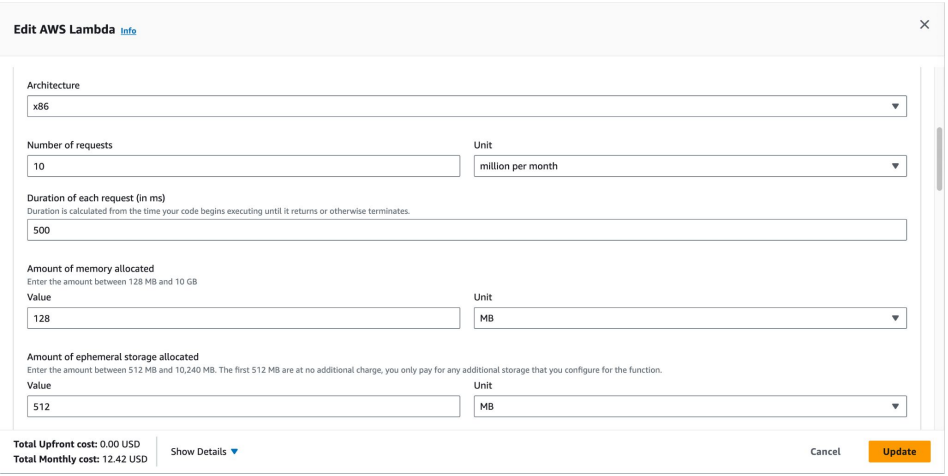
S3: 10 million request estimate

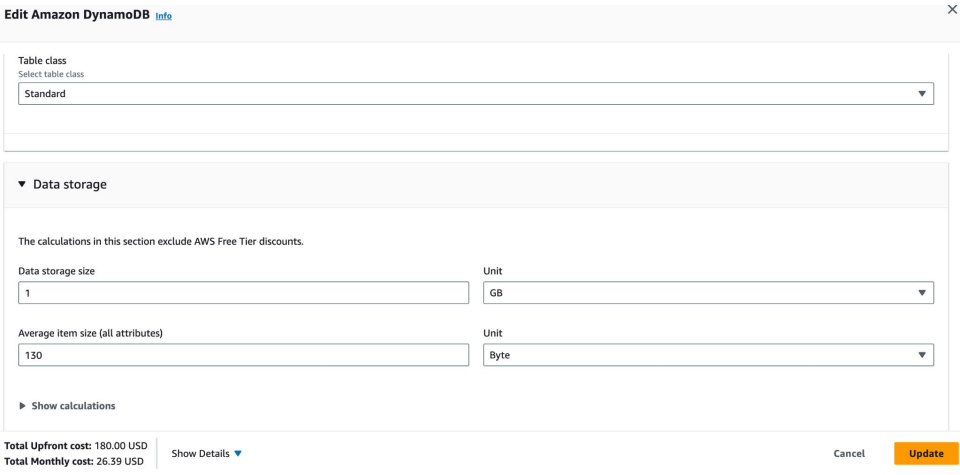


Cost Analysis by service

Lambda: 500 ms request duration

DynamoDB: 130 byte avg. item size





Cost Analysis by service

API gateway: RESTful API, 10 million request estimate