Jawad Timzit

CSS 436/Program 4: Website + Storage

2-27-2021

Report-Design

This program will use multiple services in the cloud that will work together to deploy an application in the cloud. My program will use S3 amazon storage and the NoSQL DynamoDB to load/clear/query data on the cloud through an URL that will be created when deploying the application in the cloud/Beanstalk.

The project requirements:

- ✔ The host website must perform these functions:
  - **A button call Load data:** when this is hit, the website will load data from an object that is stored at a given URL. This can be hit multiple times to load other data/overwrite and upload data that only does not exist already.
  - **A button call Clear data:** when this is hit, the S3 object will be removed and the table will be deleted from DynamoDB as well.
  - **Two input text boxes labeled First Name and Last Name:** After data is loaded. The user can type in a first and last name to search for in the database.
  - **A button called Query:** when this is hit, the results are shown on the website. The name should be exact matches. We also do not need to fill in both query boxes to query. **I can just query by entering a first name. If there is multiple have the same first name they will all be displayed.**

For this program I am using Python. I have used Flask Framework in particular. There are some imports and requirements that I needed to set up before pursuing with the implementation. For the imports like flask/urllib3/boto3 etc. I have included all those imports in the requirements.txt file that gets created with the project. So, if you like to run this program in your own machine you need to resolve all those imports. I will explain in the next session how to do this.

For the Flask web application, I also have to set the virtual environment. My visual studio 2019 prompt me to go ahead and set it up. It does it automatically otherwise you just need to set it up from the command line.

The project Flask environment set up:

- ✔ Template folder
  - Base.html
  - Load.html
  - Page1.html

- log in

✔ Application.py

- This is where I will have all the functions for this program:
  - ❖ **Query(first,last):** This function will check the DynamoDB for the first and the last name that is entered. The information that the attributes have will be displayed about this person. The program will display a message "Enter the details correctly" to the users if they enter in something that is not a name. The program also would display a message to the user if the names entered do not exist in the database. **I can just query by entering a first name. If there is multiple have the same first name they will all be displayed**
  - ❖ **Load():** when load gets invoked the database will be created and I have the awsCall() that will be used to load the data from the table that has been created in the database. The update will happen as well.
  - ❖ **Clear():** this function will clear the person's first and last name from the table in the database, and also there is a function S3Bucket_clea() that will clear the S3 object. If something went wrong, or the user is trying to clear something that does not exist, means the clear did not happen, there will be an exception that will be thrown.

✔ Requirements.txt

- This .txt file will contain all the imports and external dependencies that will be needed to build this project. I can resolve all the imports by uploading them inside this file. This file will normally contain a flask version when the FlaskWebApplication gets created. So, we need to add all other imports like boto3/urllib3…to resolve the errors those imports cause.
  - ❖ In the directory where the requirements.txt is located, we **cd** to there, and type in the command **line pip freeze > requirements.txt**
  - ❖ Once this is done, all the imports get entered inside the requirements.txt file and all the imports errors get resolved. We can always install any additional imports if anything went wrong by also using pip install package-name.

✔ .ebextensions

- Python.cnfig: contains the server environment. Responsible for executing my application.
  - ❖ File contain inside it: option_settings:
    "aws:elasticbeanstalk:container:python":
    WSGIPath: application:application

Location of the URL for my Application:

http://mywebappflaskdep-env.eba-8y2s4tpm.us-west-1.elasticbeanstalk.com

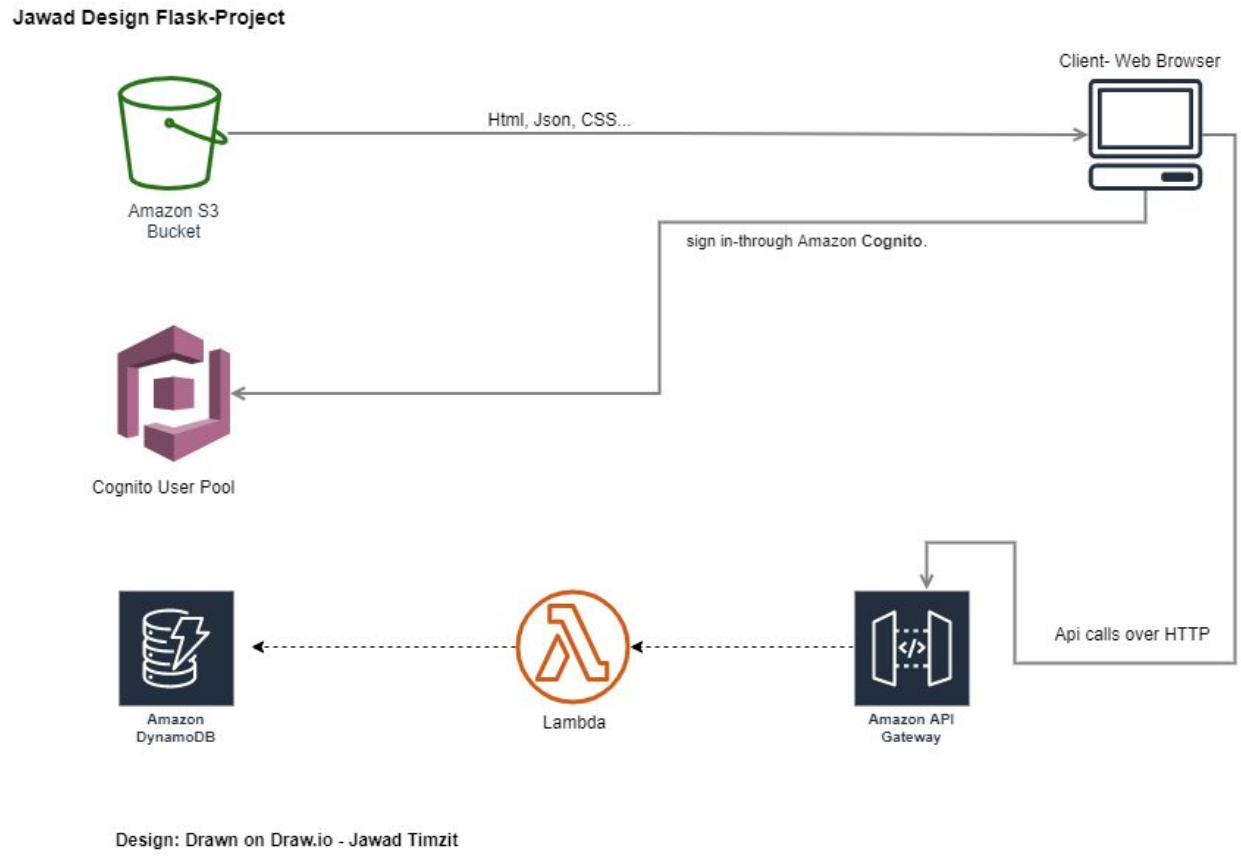
Location of where the blog / object is stored (This is public):

https://jawadcat.s3-us-west-1.amazonaws.com/folder1/input.txt

also I used this for my program to test as a target url:

http://s3-us-west-2.amazonaws.com/css490/input.txt

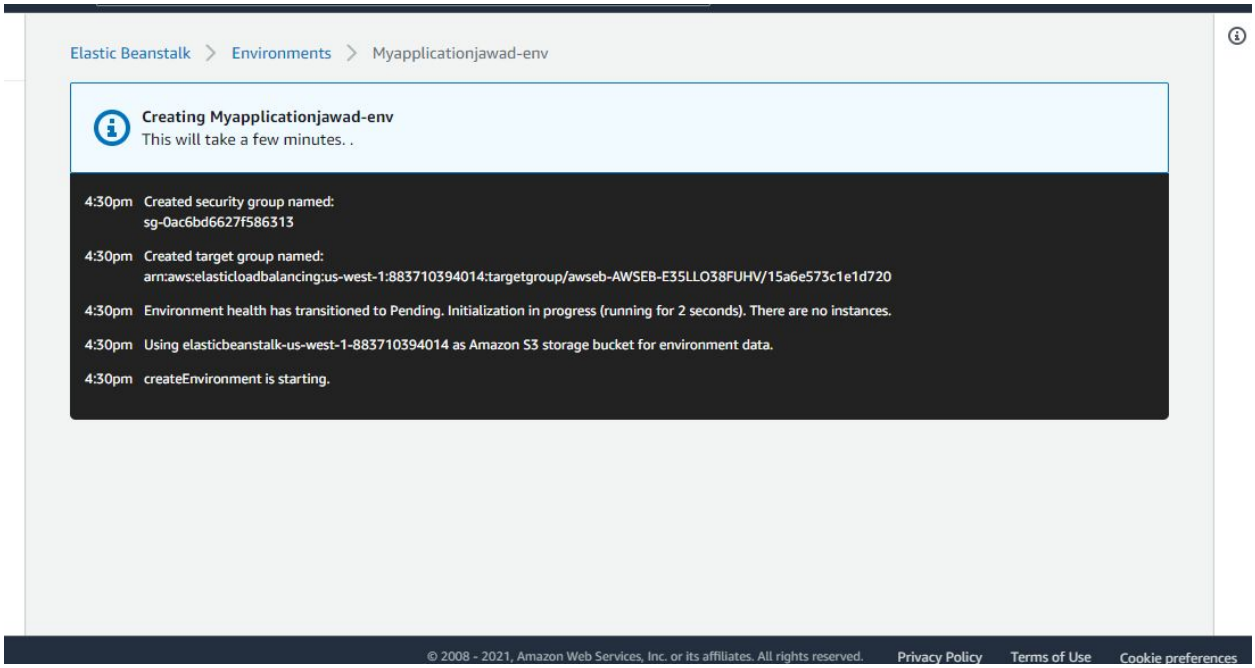
Clear design diagrams of final project:



Deployment of this the project:

- ✔ I used AWS Beanstalk to deploy/host this application.
- ✔ There are many ways we can follow to do this in Beanstalk. First, I was given so many errors, because I have hosted my application locally, but when I wanted to host in the cloud it was extremely difficult to get off errors. Since there are some requirements that I need to do before uploading my code in the Beanstalk environment.
- ✔ The zip file that was needed to upload to the Beanstalk contained:
    - application.py

- the flask object name, which we have defined in the application.py should also have the name application.
- templates folder that contains all .html files.
- static folder if we have any CSS file or any styling sheets.
- Requirements.txt file that contains all the imports and dependencies/libraries.
- .ebignore file that I have created in the same directory where my application.py exists, this file contains names of folders that we do not want to deploy/push in the cloud. For example, if we don't have any CSS file we could include a static folder here so the cloud will ignore it when we deploy the application. The main point here is we need to only upload the minimum as we can of the files since extra files that we do not need could produce errors during deployment.
- I also created a new folder called .ebextensions: inside this folder we will create a python.config that will include a server WSGIPath. This configuration file is responsible for the deployment of my application in the cloud environment. Without it I would not be able to host the application in the cloud even if it runs fine in my local machine.
- After I set up my files correctly, I can go with a few options here, like command line, but I chose to zip those files and upload the code while I create an application in Beanstalk. I just followed the steps until it was hosted (the cloud does it for you when you upload the code, we just follow the steps until it is hosted).
- Finally, the application is hosted, and I am given the URL where the application is located.



Discussion of how the site will scale with load:

Aws is distributing the incoming traffic automatically. This happens across multiple targets like Amazon EC2 instances. This can be handled in a single availability zone or multiple availability zones. Also, I can

control my compute resources by adding or removing them from the load balancer as I need. ELB can route to convenient/healthy routes if anything happens and some become unavailable.

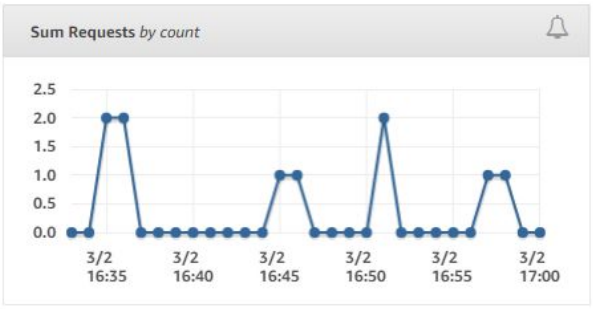
Discussion of how monitoring is done on the site:

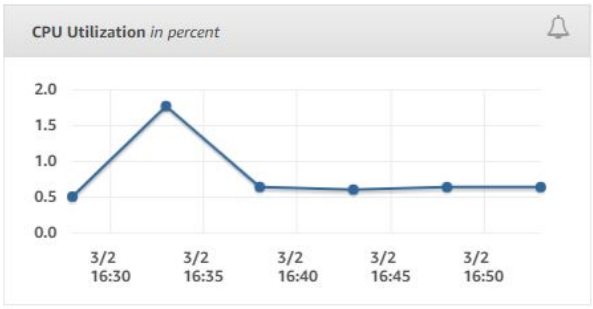
I will be monitoring the metrics that will provide me with the information about the health of the application. The metrics will be gathered from web servers and operating systems. Web servers provide information about incoming HTTP requests: how many requests came in, how many resulted in errors, and how long they took to resolve etc. elastic beanstalk provides features where I can monitor statistics about my application and create alerts that trigger when thresholds are exceeded. Elastics load balancing sends a request to each instance in an environment every 10 sends to confirm that instance is healthy.

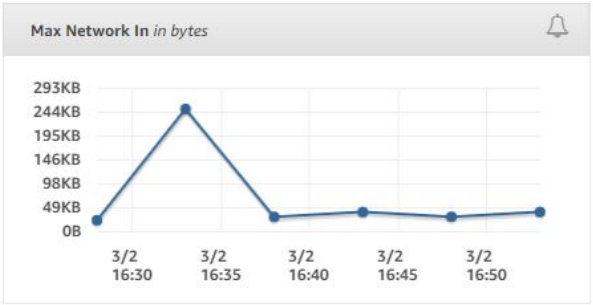
There are also the CloudWatch metrics which are used to produce graphs on the monitoring page of the environment. For example, EC2 publishes CPUUtilization, Number of bytes read and written etc.

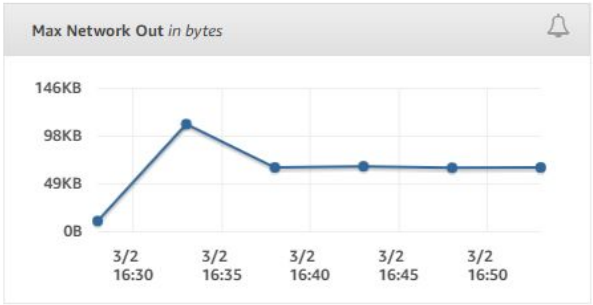
So, in conclusion to monitor the application health:

- On the application environment tab, we click **Motoring**. This monitoring panel includes a set of graphs showing resource usage for my application environment.









Estimate of your SLA:

	Amazon S3	EC2	DynamoDB	Overall
Availability %	99.95	99.99	99.99	99.95*99.99*99.99 = 99.93
Durability %	99.999999999	99.999999999	100	9999999999*9999999999*100 = 99.9999998