CS 218, Spring 2019, Project Presentation

Image Portal Web Application using Amazon Web Services

Presented by:
Team 6
Abhishek Manoj Sharma
Anand Vishwakarma
Rajeshwari Deepak Chandratre

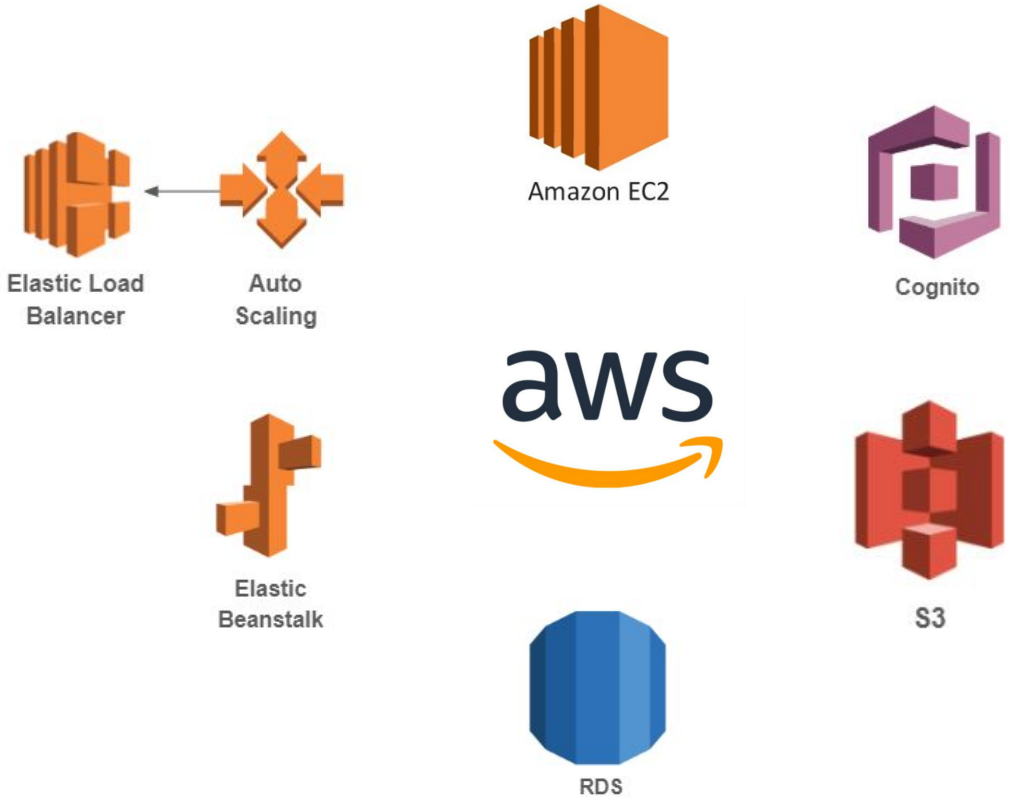
Introduction

An image portal where users can upload, view / download, and search for images.

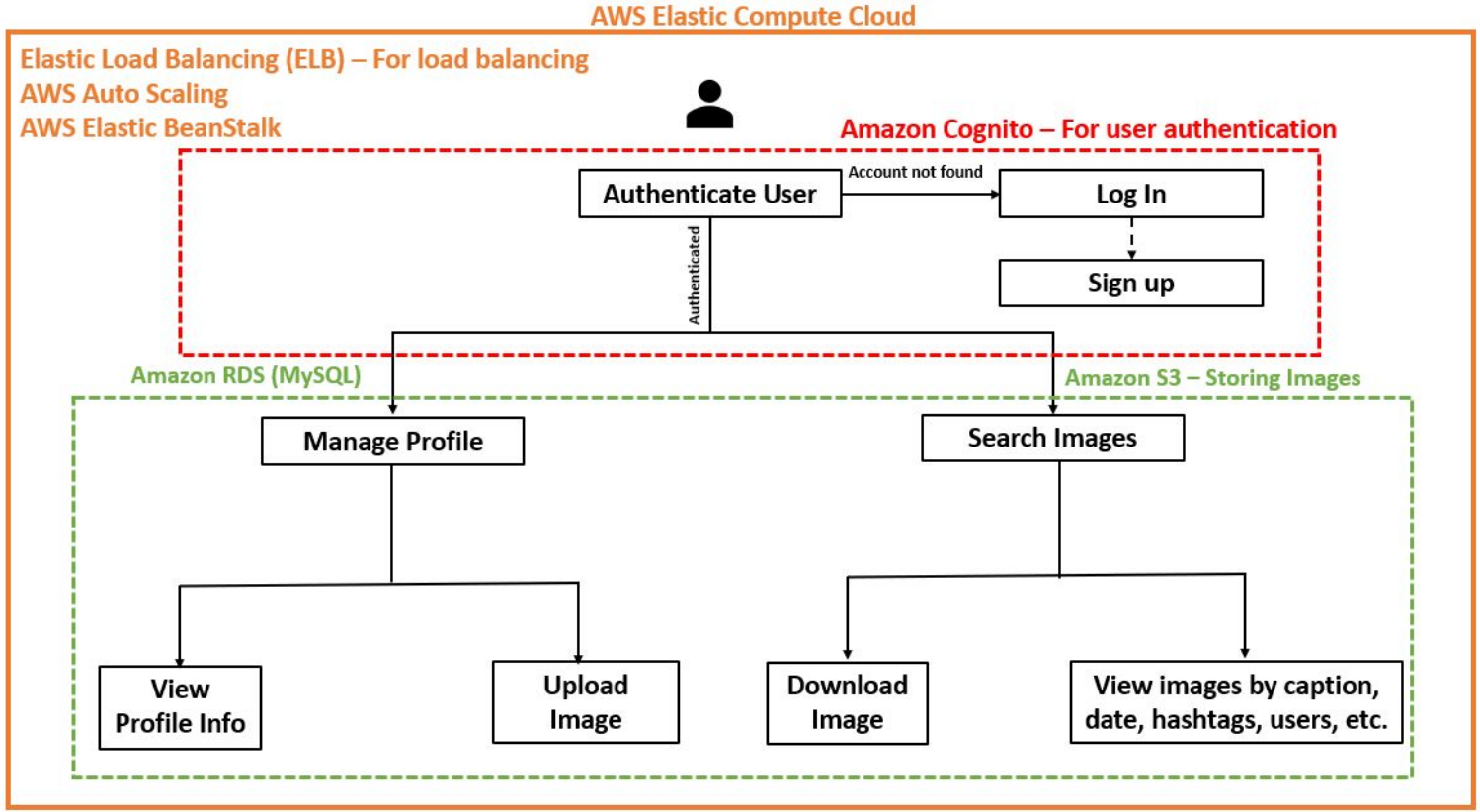
Functions

- User Sign Up
- User Verification
- User Login
- User Session
- Image Upload
- Image Download
- Image Captioning
- Image Hashtags
- Search by Keyword
- Search by Username
- Search by Hashtags
- View User Profile

Cloud Components Used

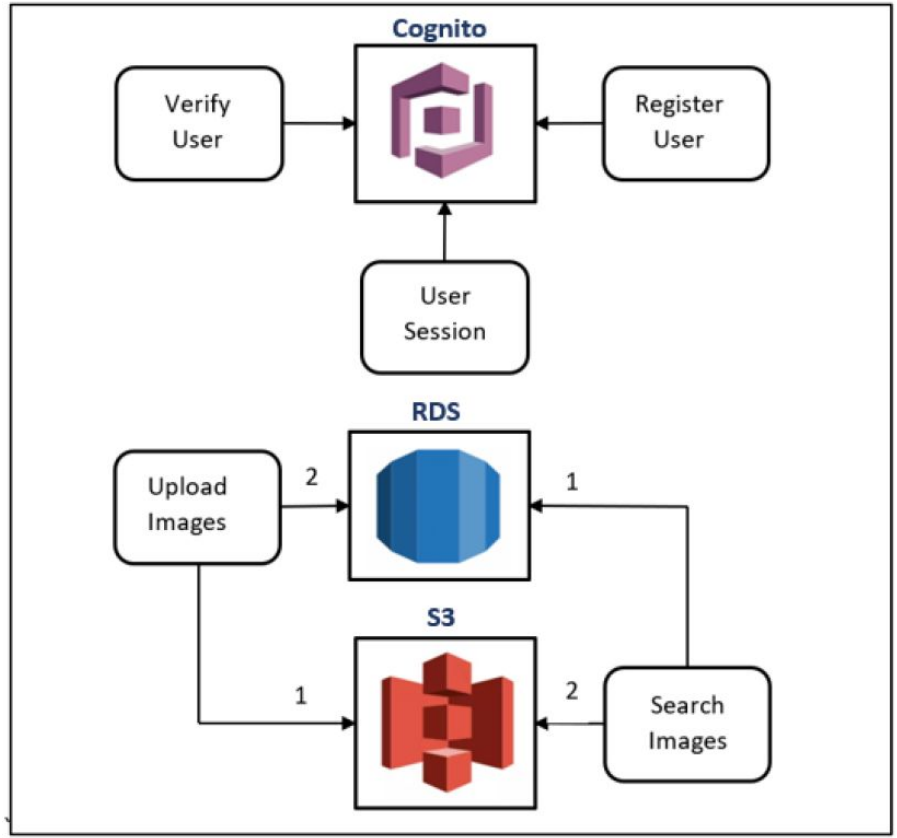


Architecture



Data Flow

- Cognito is used to manage user signups, logins and sessions.
- All images are stored on S3, and a record for each image upload is created in a table in RDS.
- RDS (MySQL) stores image upload data containing the username, caption, image S3 URL, and timestamp.



Hosting Application

Elastic Compute Cloud (EC2)

- The application is hosted on an Amazon EC2 instance.
- The Operating System is based on a Linux 2.8.3 64-bit system.
- The instance also contains Python 3.6 (64-bit) for the Python application server built using Flask.

User Management

Cognito

- Cognito is used for user management and tracking.
- We have used Amazon Cognito’s JS SDK to facilitate the user management activities and session tracking.
- User activities include: Signups, verification (via email), logins, and session management.

Storage

Simple Storage Server (S3)

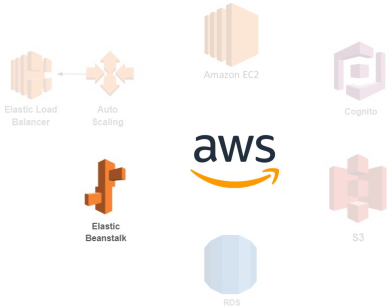
- Application contains several writes: Leveraged S3 from AWS for storing images.
- We have used the Boto3 Python package for this.
- Each image is stored in the S3 bucket using a unique name (timestamp suffix).
- Based on this unique name, a unique URL is generated by S3 which is stored as a reference in our database.

Database

Relational Database Service

- MySQL based RDS server.
- We have used the PyMySQL Python package for this.
- Image uploads: When a new image is uploaded, a corresponding record is added.
- Image retrieval: When images are retrieved for the home feed or a specific search, the corresponding SQL query is executed to retrieve and display the results.

Deployment



Elastic Beanstalk

- We have used Elastic Beanstalk to deploy our application.
- Beanstalk contains a predefined Python environment which can be leveraged to host Python (3.x) based web applications.
- It also has a seamless integration with Flask based applications (zip upload).
- Beanstalk also simplifies auto scaling, load balancing, and health monitoring.

Availability and Auto Scaling

AWS Auto Scaling

- Auto Scaling adds or removes new EC2 instances based on the configured triggers.
- High availability achieved by setting the minimum instances to 2.
- ‘Network In’ metric selected as our application involves several image uploads.
- For demo purpose: Upper threshold 1 mb, lower threshold 500 kb.

Load Balancing

Elastic Load Balancer

- The load balancer continuously monitors the incoming traffic.
- Distributes it to the instances used by our application.
- Performs health checking at a regular interval to determine the health of our instances.

Demo