Image Portal Web Application using Amazon Web Services
Project Report

Presented to
Dr. Kong Li

Department of Computer Science
San José State University

In Partial Fulfillment
Of the Requirements for the Class
CS 218, Topics in Cloud Computing, Spring 2019

By
Team # 6
Abhishek Manoj Sharma
Anand Vishwakarma
Rajeshwari Deepak Chandratre

May 5, 2019

Table of Contents

Intended Users or Service Consumers

As stated in the proposal, the intended users for this service are people who are searching for images uploaded by other people to use for their own personal or commercial use. Once a user opts to share an image on the portal, it is made available to all registered users.

List of functionalities and operations

- User signup (Cognito)
- User verification (Cognito)
- User login (Cognito)
- Image upload (RDS and S3)
- View profile (RDS and S3)
- Search images by users (RDS and S3)
- Search images by keywords (RDS and S3)

Major areas, components, tasks

Task	**Team Members Responsible**	**Completion Date**
Researching on the cloud service to be used for the project	Abhishek, Anand, Rajeshwari	March 5, 2019
Identifying cloud services to be used for the project	Abhishek, Anand, Rajeshwari	March 9, 2019
Configuring and setting up Cognito user pool	Abhishek, Rajeshwari	April 21, 2019
Implementing user signups, validations, authentications, logins and sessions using Cognito and JavaScript	Abhishek, Rajeshwari	April 28, 2019
Implementing the database design and configuring the MySQL based RDS instance	Anand, Rajeshwari	April 25, 2019

Configuring the S3 bucket for storing images	Abhishek	April 27, 2019
Implementing application logic for storing images in S3	Abhishek, Anand	April 30, 2019
Implementing application logic for writing the image details to RDS	Abhishek, Anand	May 1, 2019
Implementing the application logic for image upload and handling it using Python (Flask) and HTML	Abhishek	May 1, 2019
Configuring the application and project for Elastic Beanstalk (EBS) deployment	Abhishek, Anand	May 2, 2019
Configuring the EBS app environment, and installing the application	Abhishek, Anand	May 2, 2019
Researching on the parameters to be used for auto scaling and load balancers	Abhishek, Rajeshwari	May 3, 2019
Configuring auto scaling, load balancers, and health checks for the application	Abhishek, Rajeshwari	May 3, 2019

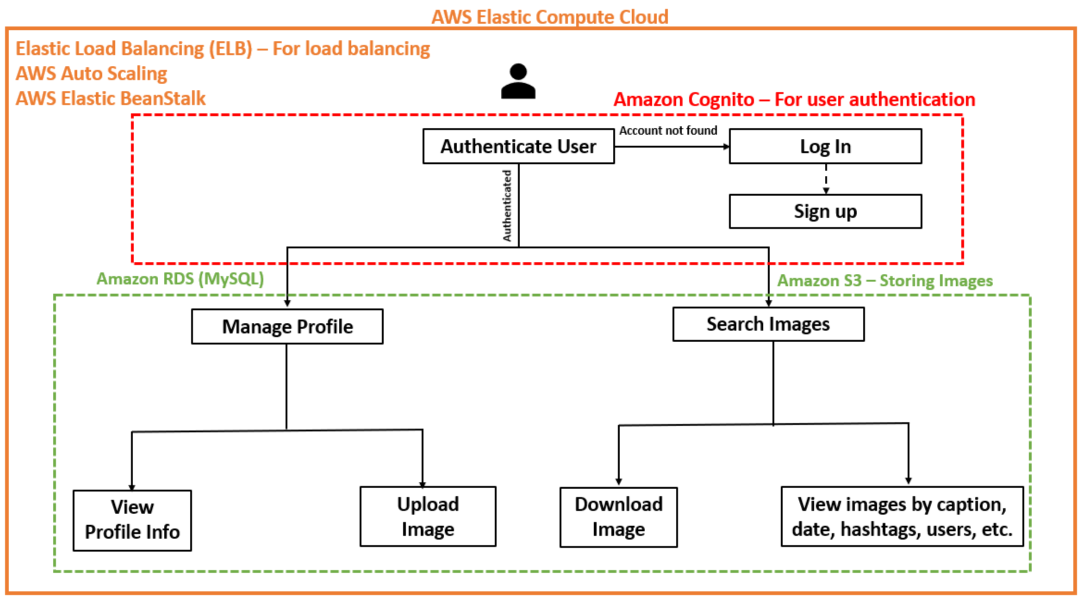
Project URL

The project is deployed on Amazon Web Services (AWS) via Elastic Beanstalk (EBS) on the us-east-2 region. The link for the project is as follows:

http://cs218imageportal.us-east-2.elasticbeanstalk.com

Design

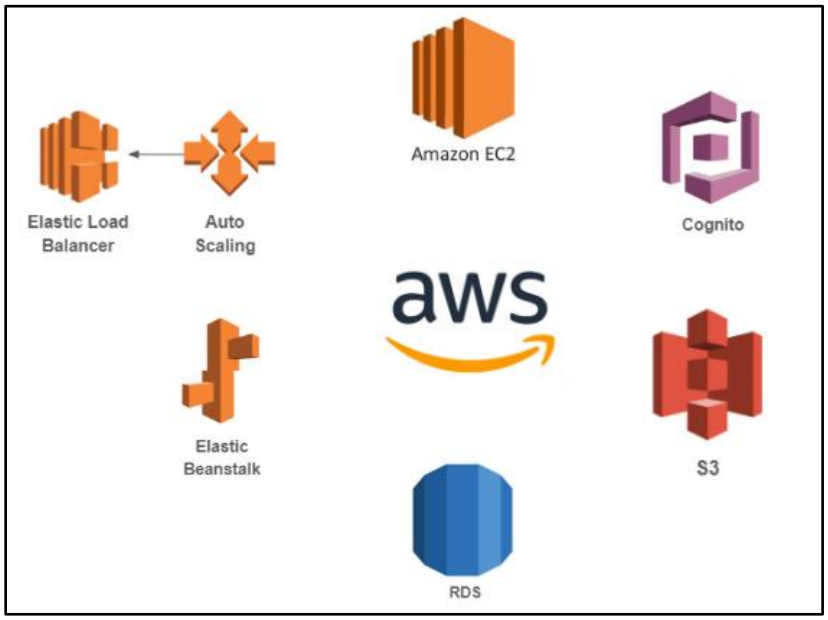
Architecture


AWS Elastic Compute Cloud
Elastic Load Balancing (ELB) – For load balancing
AWS Auto Scaling
AWS Elastic BeanStalk
Amazon Cognito – For user authentication
Authenticate User
Account not found
Log In
Sign up
Authenticated
Amazon RDS (MySQL)
Amazon S3 – Storing Images
Manage Profile
Search Images
View Profile Info
Upload Image
Download Image
View images by caption, date, hashtags, users, etc.

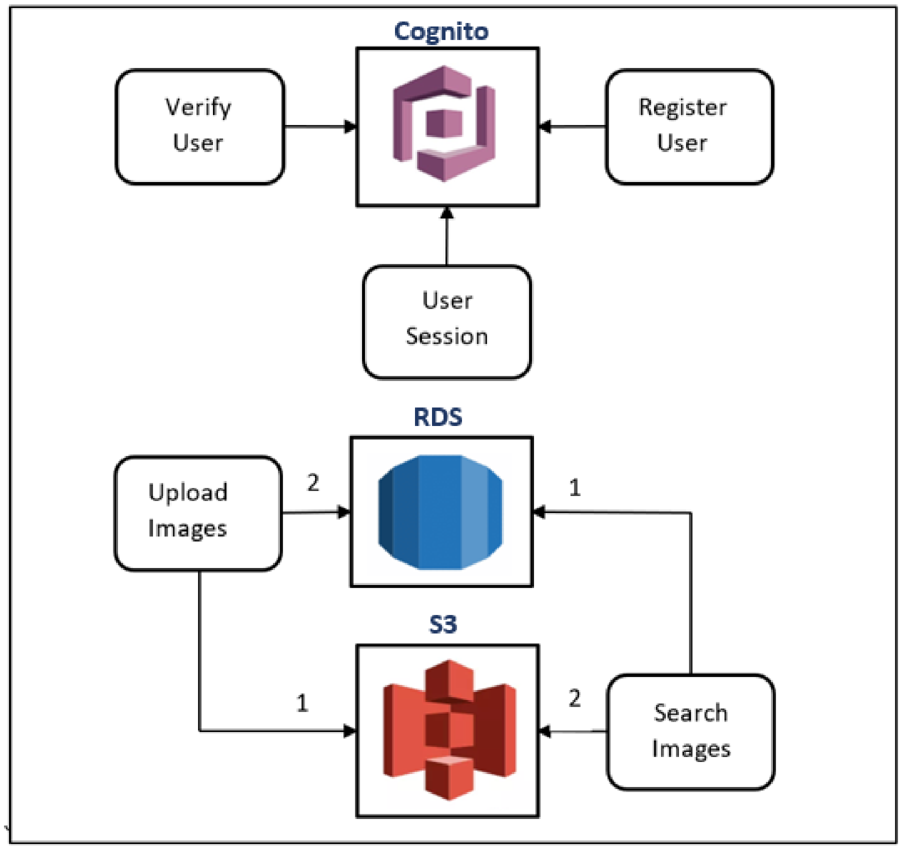
Components

Following is the list of AWS components we have used in this project:

1. Amazon Elastic Compute Cloud (EC2)
2. Amazon Cognito
3. Amazon Simple Storage Server (S3)
4. Amazon Relational Database Services (RDS)
5. AWS Elastic Beanstalk
6. AWS Auto Scaling
7. Amazon Elastic Load Balancing



Data Flow Diagram



Amazon Cognito takes care of the user management for user sign ups, logins, session management and sign outs. After the user logs in, images can be viewed, searched and uploaded. This is achieved with the help of Amazon S3 and Amazon MySQL based RDS. 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. While searching the images, RDS is queried based on different conditions, and the with the help of the corresponding record in the RDS, images are fetched from S3 bucket and rendered.

Load Balancing, High Availability, Scalability

For load balancing, the application uses the Elastic Load Balancing service which uses a load balancer created by the AWS Elastic Beanstalk. Since the application involves several image uploads, ELB takes care of load balancing by continuously monitoring incoming traffic. The load balancer distributes the traffic to the instances used by our application. Our application is using a Classic Load Balancer that basically enables routing of requests such as HTTP, HTTPS or TCP traffic to ports on our application’s instances’. We've configured our load balancer with a standard web server on port 80.

The load balancer also performs health checking to determine the health of the EC2 instances running our application. At a regular interval of 10 seconds, the health check feature makes a request to our specified URL ‘/check-status’. If this URL returns an error message or fails to respond within a timeout period of 5 seconds, it declares the instance as ‘unhealthy’ meaning that the health check has failed.

High availability is achieved by keeping the minimum number of instances that will always be present to 2. Both the EC2 instances are present in different availability zones than each other. This ensures that even if one of the instances is down due to any reason, the other instance will be available in the different availability zone, thus avoiding any downtime.

Scalability is achieved by using AWS Auto Scaling feature of AWS Elastic Beanstalk which spawns or removes new EC2 instance as and when the load increases or decreases, thus achieving scalability.

For this project, we have set the minimum number of available instances as 2 and maximum number of instances to 4. It means that at any point of time, there will always be at least one instance available, but the maximum number of instances that could be present will be 4. This scaling of instances will be dependent on the incoming network load to our application. In this application, the incoming network maximum load is set to 1 MB and minimum is set to 0.5 MB. These values act as upper and lower thresholds respectively. As soon as the network load increases above maximum threshold, a new EC2 instance is created to achieve high availability and any subsequent request is distributed to the instances. On the same path, if the load goes below minimum threshold, one EC2 instance is terminated, thus scaling in.

Project Implementation

OS, Languages, Platform, Technologies, and Frameworks

The application is deployed via Elastic Beanstalk and is running on an Amazon EC2 instance based on Linux 2.8.3 64-bit. The languages used for the implementation are Python, JavaScript, AJAX, HTML, and CSS. The frameworks we have used are Flask (for Python server), Boto3 (to communicate with S3 via Python), PyMySQL (to communicate with MySQL based RDS server), Cognito SDK (for JavaScript), and Bootstrap for HTML components.

Cloud Features Leveraged

AWS Elastic Beanstalk is a service provided by AWS for handling the deployment of the applications. It also takes care of provisioning of capacity, auto-scaling, load balancing, and application health monitoring. After simply uploading our application code, the Elastic Beanstalk deploys the application and handles scaling of the application based on triggers. 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). Hence, we used Beanstalk for deployment of our application. We also leveraged the auto-scaling, load balancing, and health monitoring features of AWS Elastic Beanstalk. As mentioned earlier, Auto Scaling adds or removes new EC2 instances based on the configured triggers. The load balancing feature of Beanstalk automatically monitors the incoming traffic continuously and distributes them to the EC2 instances in use by our application. It performs health checking at a regular interval to determine the health of our instances. With the help of these auto-scaling and load balancing features of AWS Elastic Beanstalk, we are able to achieve Load Balancing, High Availability, and Scalability in our application, details of which have been already explained before.

Amazon Elastic Compute Cloud (EC2) is a web service that provides resizable compute capacity in the cloud. For hosting our application, we have used Amazon EC2 instance. This instance, as mentioned earlier, a Linux 2.8.3 64-bit operating system. The EC2 instance also contains Python 3.6 (64-bit) for the Python application server built using Flask.

Amazon Cognito is a user management service provided by AWS. It provides user sign-up, sign-in, and access control to the web and mobile applications. AWS Cognito maintains Amazon Cognito User Pools. These User Pools provide a secure user directory that scales to millions of users. We can also customize the user sign-up based details as per the need of our application. We have made use of Amazon Cognito for user management and tracking in our application. In our project, the user related activities include user sign-up, verification via email, logins, and session management. A user can sign up to the image portal using username, password, and email ID which is stored in Amazon Cognito User Pool. The user verification takes place using the email address of the user, the Cognito service sends a unique code to the user’s email address, and the user is able to complete the sign-up process only after he/she enters this code while signing-up. Logging in to the portal is username and password based. We have used

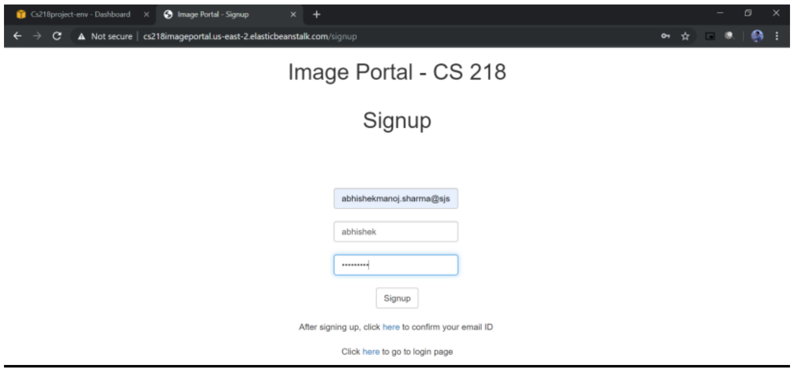
Amazon Cognito’s JS SDK to facilitate the user management activities and session tracking. The logout functionality is also implemented with the help of Cognito.

Amazon provides an object storage service called as Amazon Simple Storage Service (Amazon S3) which enables the customers to store any amount of data as per the need of the user application. Amazon S3 provides easy management and access control of the stored data to meet the business requirement. We are leveraging Amazon S3 to store our database of images as our application contains several writes. In our application, 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. We have used the Boto3 library to interact with our S3 bucket.

To handle backend related activities, Amazon provides a relational database management service called as Amazon Relational Database Service (Amazon RDS). For this project, we use the MySQL variant of RDS. With the aid of Amazon RDS, we can use web-services to create, manage, and delete relational databases through our applications. In our application, we have used MySQL based RDS instance. We have used the PyMySQL Python package for this. It is an open-source Python library that connects to MySQL from Python. In our project, RDS stores image upload data containing the username, caption, image S3 URL, and timestamp. Every time a new image is uploaded, a new record is created in RDS with the correct S3 link and all other image and user related details as mentioned before. Search functionality is also implemented with the help of RDS, where the user can search image using different categories such as by keyword, username, hashtag, and profile. When images are retrieved for the home feed or a specific search query, the corresponding SQL query is executed to retrieve and display the results.

Sample Execution

1. User signup:



Cs218project-env - Dashboard | Image Portal - Signup

Not secure | cs218imageportal.us-east-2.elasticbeanstalk.com/signup

Image Portal - CS 218

Signup

abhishekmanoj.sharma@sjs

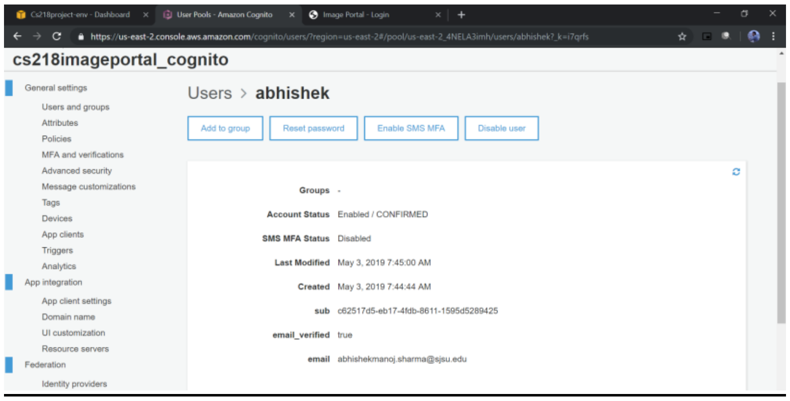
abhishek

•••••••••

Signup

After signing up, click here to confirm your email ID

Click here to go to login page



Cs218project-env - Dashboard | User Pools - Amazon Cognito | Image Portal - Login

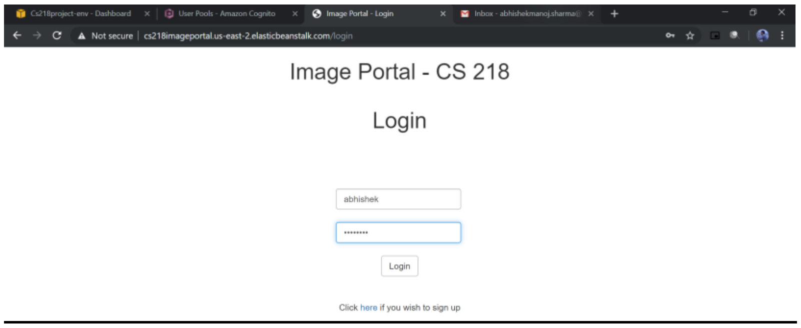
https://us-east-2.console.aws.amazon.com/cognito/users/?region=us-east-2#/pool/us-east-2_4NELA3imh/users/abhishek?_k=i7qrfs

cs218imageportal_cognito

Users > abhishek

Groups	-
Account Status	Enabled / CONFIRMED
SMS MFA Status	Disabled
Last Modified	May 3, 2019 7:45:00 AM
Created	May 3, 2019 7:44:44 AM
sub	c62517d5-eb17-4fdb-8611-1595d5289425
email_verified	true
email	abhishekmanoj.sharma@sjsu.edu

2: User login



Cs218project-env - Dashboard | User Pools - Amazon Cognito | Image Portal - Login | Inbox - abhishekmanoj.sharma@

Not secure | cs218imageportal.us-east-2.elasticbeanstalk.com/login

Image Portal - CS 218

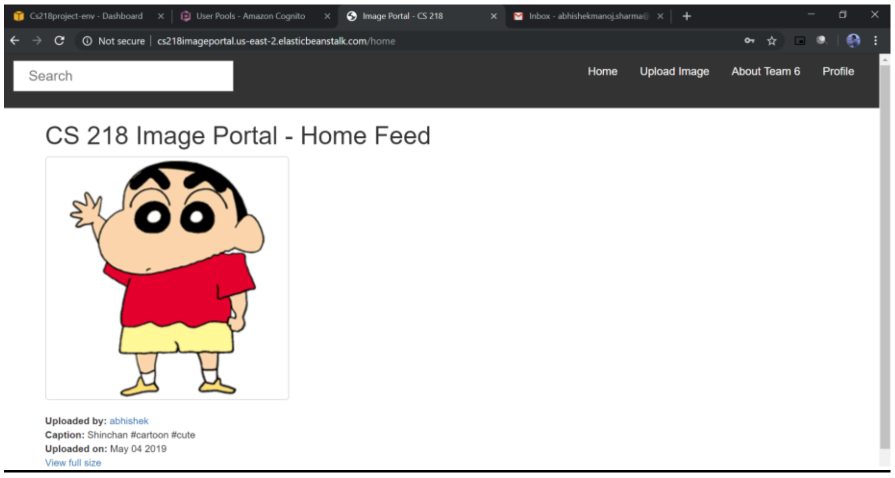
Login

abhishek

••••••••

Login

Click here if you wish to sign up


Cs218project-env - Dashboard
User Pools - Amazon Cognito
Image Portal - CS 218
Inbox - abhishekmanoj.sharma@
Not secure | cs218imageportal.us-east-2.elasticbeanstalk.com/home
Search
Home
Upload Image
About Team 6
Profile

CS 218 Image Portal - Home Feed

Uploaded by: abhishek
Caption: Shinchan #cartoon #cute
Uploaded on: May 04 2019
View full size

3: Upload Image:

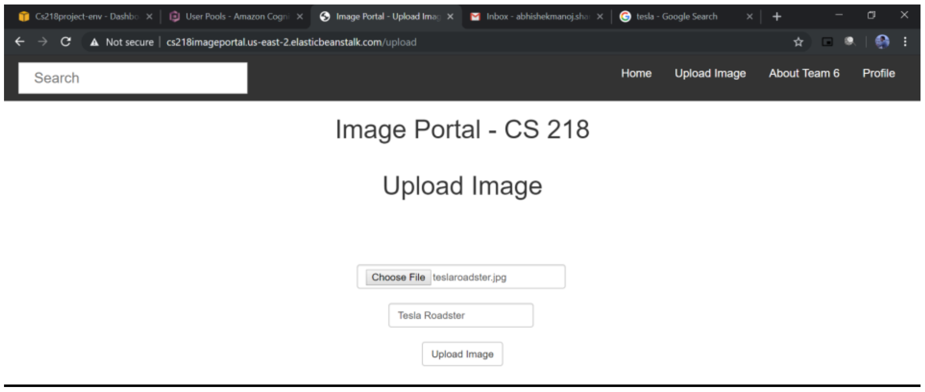

Cs218project-env - Dashbo
User Pools - Amazon Cogn
Image Portal - Upload Ima
Inbox - abhishekmanoj.sha
tesla - Google Search
Not secure | cs218imageportal.us-east-2.elasticbeanstalk.com/upload
Search
Home
Upload Image
About Team 6
Profile

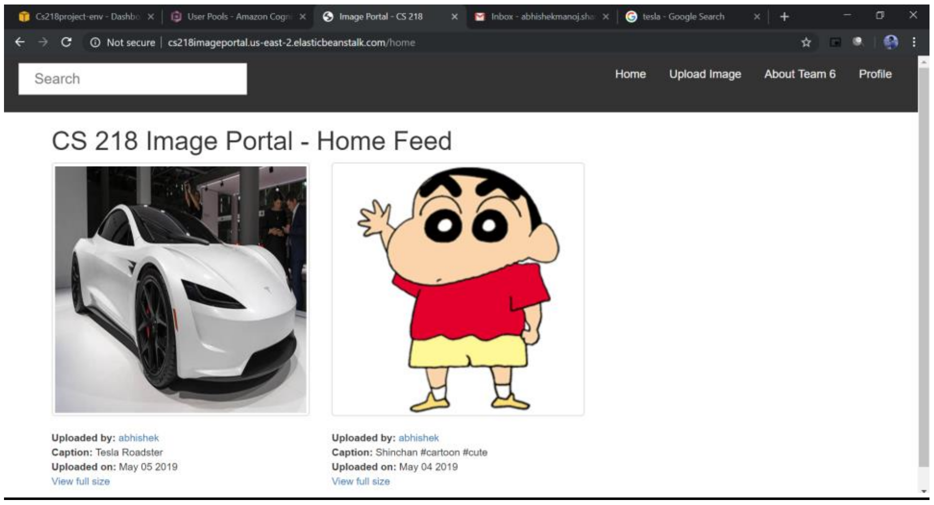
Image Portal - CS 218

Upload Image

Choose File teslaroadster.jpg

Tesla Roadster

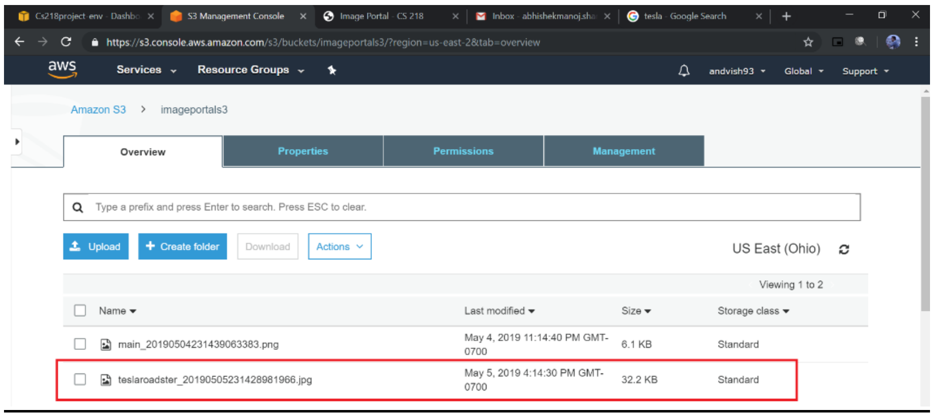
Upload Image


Cs218project-env - Dashbo
User Pools - Amazon Cogn
Image Portal - CS 218
Inbox - abhishekmanoj.sha
tesla - Google Search
Not secure | cs218imageportal.us-east-2.elasticbeanstalk.com/home
Search
Home
Upload Image
About Team 6
Profile

CS 218 Image Portal - Home Feed

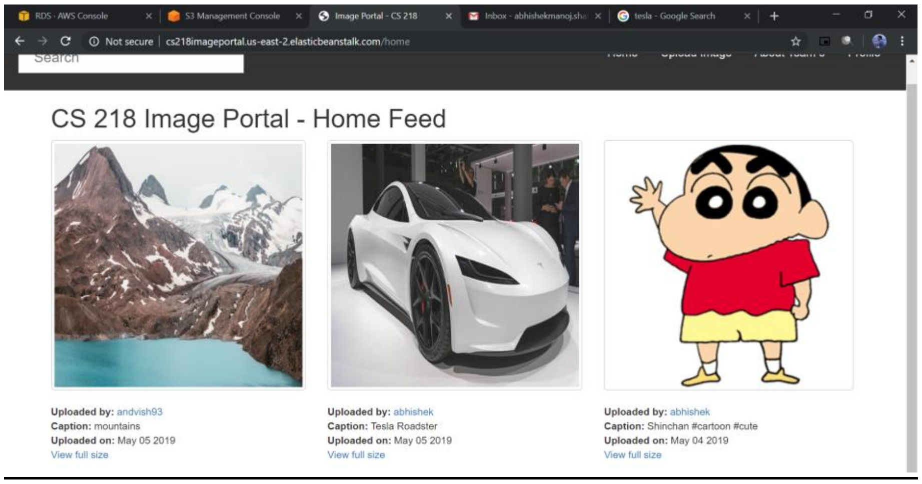
Uploaded by: abhishek
Caption: Tesla Roadster
Uploaded on: May 05 2019
View full size

Uploaded by: abhishek
Caption: Shinchan #cartoon #cute
Uploaded on: May 04 2019
View full size



Name	Last modified	Size	Storage class
main_20190504231439063383.png	May 4, 2019 11:14:40 PM GMT-0700	6.1 KB	Standard
teslaroadster_20190505231428981966.jpg	May 5, 2019 4:14:30 PM GMT-0700	32.2 KB	Standard

4: Search by user:

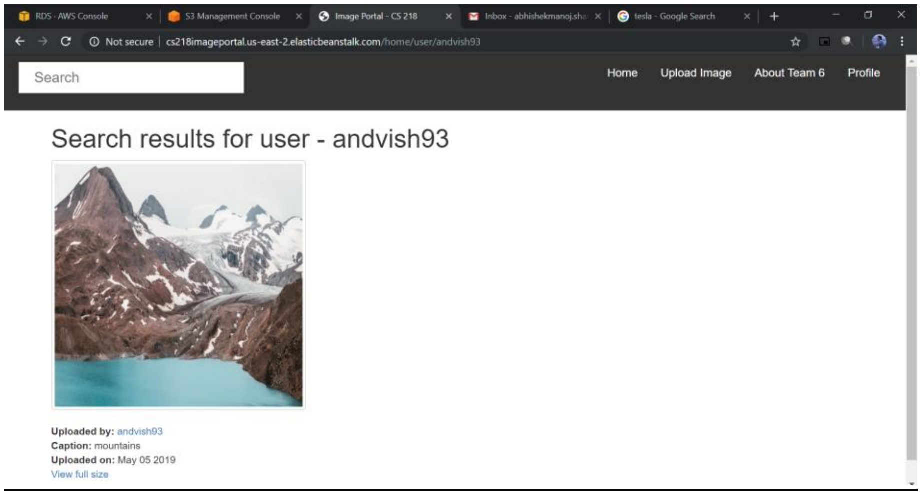


CS 218 Image Portal - Home Feed

Uploaded by: andvish93
Caption: mountains
Uploaded on: May 05 2019
View full size

Uploaded by: abhishek
Caption: Tesla Roadster
Uploaded on: May 05 2019
View full size

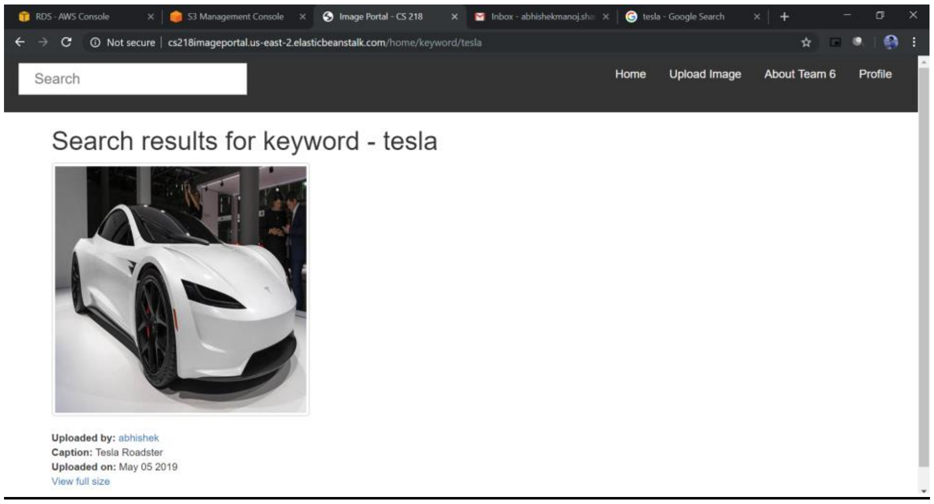
Uploaded by: abhishek
Caption: Shinchan #cartoon #cute
Uploaded on: May 04 2019
View full size



Search results for user - andvish93

Uploaded by: andvish93
Caption: mountains
Uploaded on: May 05 2019
View full size

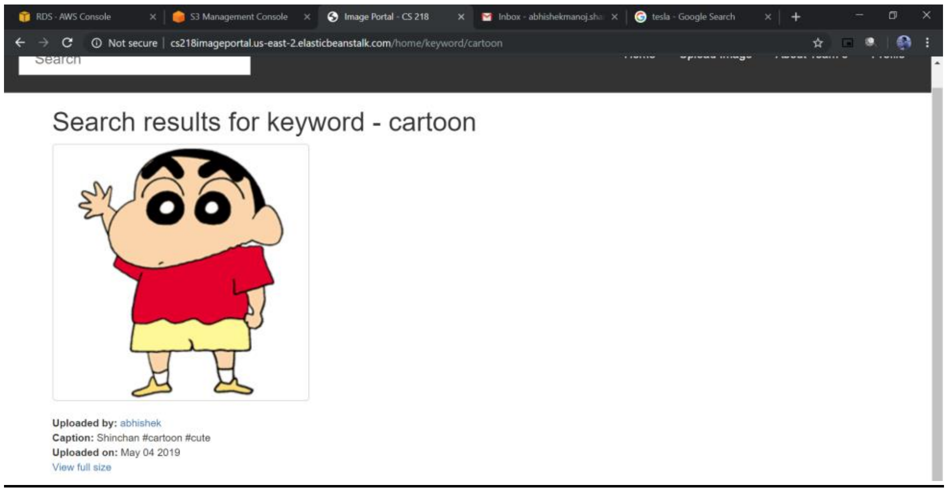
5: Search by keyword / hashtag:



cs218imageportal.us-east-2.elasticbeanstalk.com/home/keyword/tesla

Search results for keyword - tesla

Uploaded by: abhishek
Caption: Tesla Roadster
Uploaded on: May 05 2019
View full size

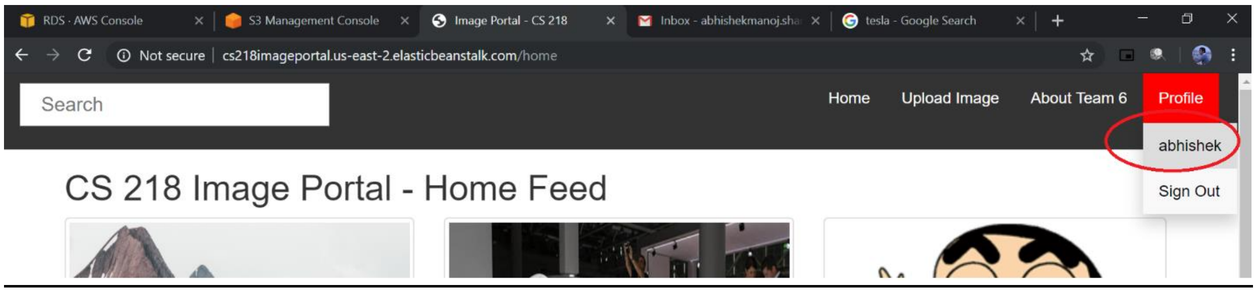


cs218imageportal.us-east-2.elasticbeanstalk.com/home/keyword/cartoon

Search results for keyword - cartoon

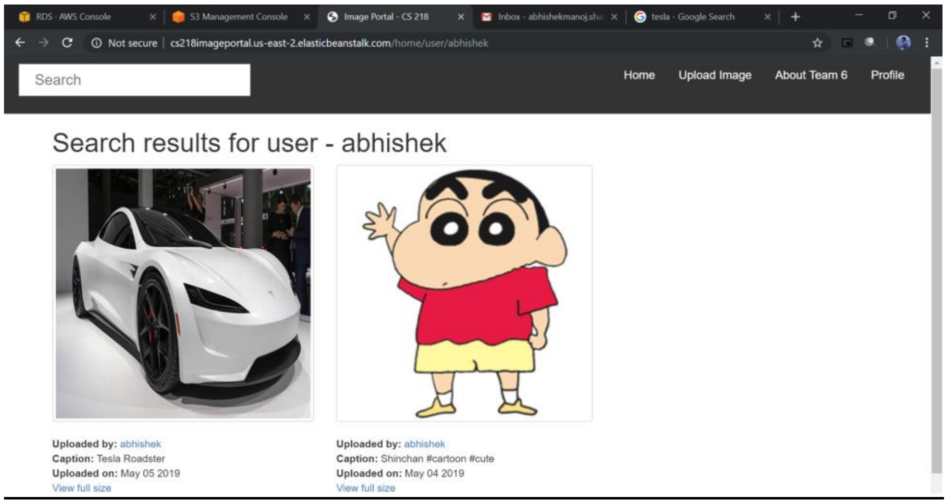
Uploaded by: abhishek
Caption: Shinchan #cartoon #cute
Uploaded on: May 04 2019
View full size

6: Profile Page:



cs218imageportal.us-east-2.elasticbeanstalk.com/home

CS 218 Image Portal - Home Feed

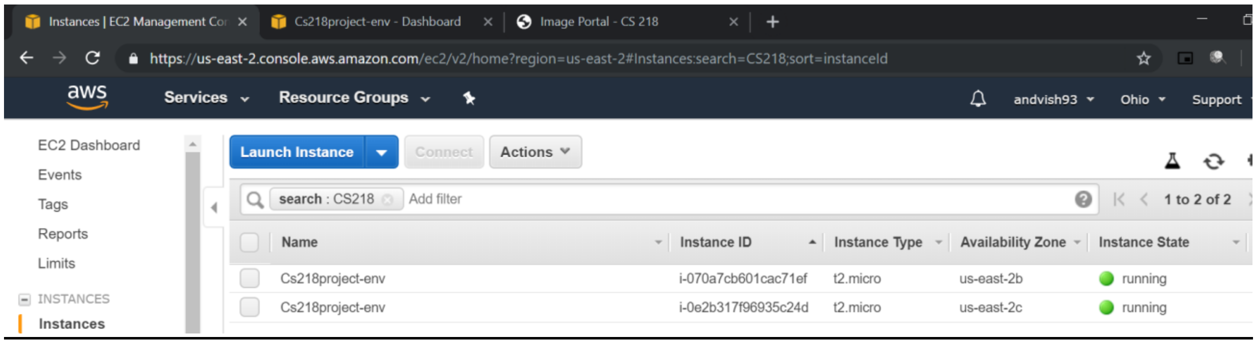


Search results for user - abhishek

Uploaded by: abhishek
Caption: Tesla Roadster
Uploaded on: May 05 2019
View full size

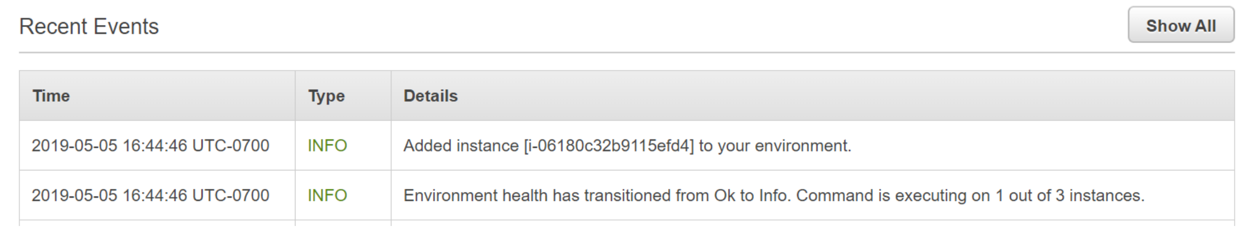
Uploaded by: abhishek
Caption: Shinchan #cartoon #cute
Uploaded on: May 04 2019
View full size

7: Normal Load – 2 instances:



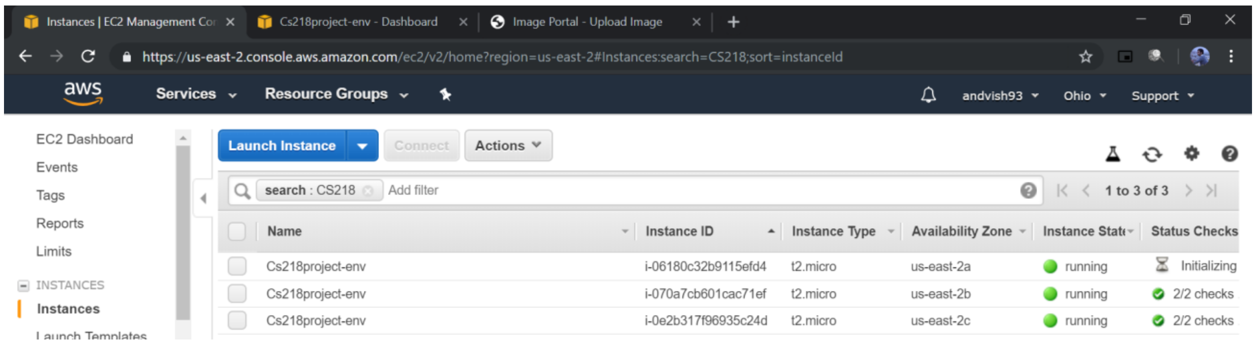
Name	Instance ID	Instance Type	Availability Zone	Instance State
Cs218project-env	i-070a7cb601cac71ef	t2.micro	us-east-2b	● running
Cs218project-env	i-0e2b317f96935c24d	t2.micro	us-east-2c	● running

8: Heavy Load – 3 instances:

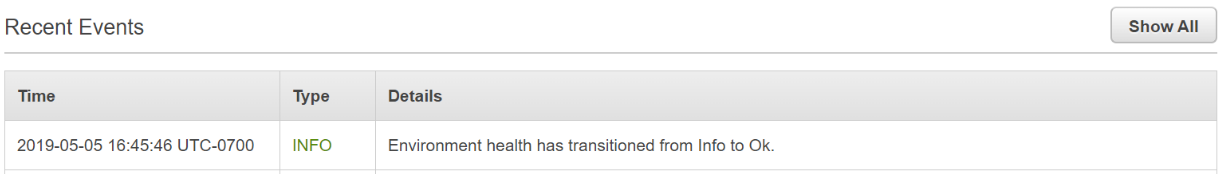


Recent Events

Time	Type	Details
2019-05-05 16:44:46 UTC-0700	INFO	Added instance [i-06180c32b9115efd4] to your environment.
2019-05-05 16:44:46 UTC-0700	INFO	Environment health has transitioned from Ok to Info. Command is executing on 1 out of 3 instances.

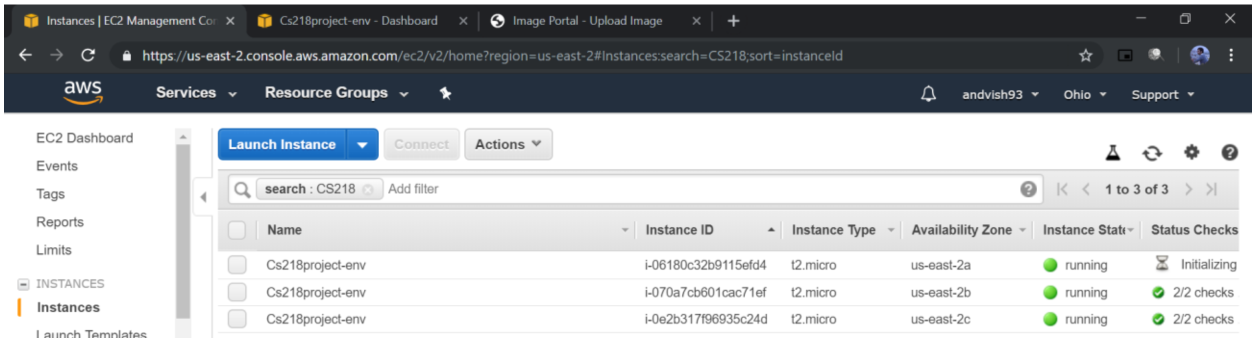

Instances | EC2 Management Co
Cs218project-env - Dashboard
Image Portal - Upload Image
https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Instances:search=CS218;sort=instanceId
aws
Services
Resource Groups
andvish93
Ohio
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Launch Instance
Connect
Actions
search : CS218
Add filter
1 to 3 of 3

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
Cs218project-env	i-06180c32b9115efd4	t2.micro	us-east-2a	● running	Initializing
Cs218project-env	i-070a7cb601cac71ef	t2.micro	us-east-2b	● running	✓ 2/2 checks
Cs218project-env	i-0e2b317f96935c24d	t2.micro	us-east-2c	● running	✓ 2/2 checks



Time	Type	Details
2019-05-05 16:45:46 UTC-0700	INFO	Environment health has transitioned from Info to Ok.

9. Load Balancer:


Instances | EC2 Management Co
Cs218project-env - Dashboard
Image Portal - Upload Image
https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Instances:search=CS218;sort=instanceId
aws
Services
Resource Groups
andvish93
Ohio
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Launch Instance
Connect
Actions
search : CS218
Add filter
1 to 3 of 3

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
Cs218project-env	i-06180c32b9115efd4	t2.micro	us-east-2a	running	Initializing
Cs218project-env	i-070a7cb601cac71ef	t2.micro	us-east-2b	running	2/2 checks
Cs218project-env	i-0e2b317f96935c24d	t2.micro	us-east-2c	running	2/2 checks

All three instances are created in different availability zones.

Database Table Schema

The RDS database consists a table to maintain all image uploads. It consists information about the user who uploaded the photo, S3's URL for the image, image caption, and the image upload date.

Create Table Script:

```
CREATE TABLE imageportaldb.uploads (Username VARCHAR(255), Image_Caption VARCHAR(1000), Image_URL VARCHAR(1000), Upload_Date VARCHAR(100))
```

Sample Record:

User	Image Caption	Image_URL	Upload_Date
Abhishek	Tesla Roadster	https://s3.us-east-2.amazonaws.com/imageportals3/teslaroadster_20190505231428981966.jpg	20190505231428981966

Sample Text Based Log Files

The log files created by AWS Beanstalk are provided as separate files. Below is a sample of the log:

```
172.31.14.254 (-) - - [05/May/2019:23:21:57 +0000] "GET /check-status HTTP/1.1" 200 43 "-" "ELB-HealthChecker/1.0"
172.31.27.181 (-) - - [05/May/2019:23:21:59 +0000] "GET /check-status HTTP/1.1" 200 43 "-" "ELB-HealthChecker/1.0"
172.31.14.254 (-) - - [05/May/2019:23:22:07 +0000] "GET /check-status HTTP/1.1" 200 43 "-" "ELB-HealthChecker/1.0"
172.31.14.254 (24.4.149.128) - - [05/May/2019:23:22:07 +0000] "GET /home/user/abhishek HTTP/1.1" 200 6736 "http://cs218imageportal.us-east-2.elasticbeanstalk.com/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36"
172.31.27.181 (-) - - [05/May/2019:23:22:09 +0000] "GET /check-status HTTP/1.1" 200 43 "-" "ELB-HealthChecker/1.0"
```

Major modifications from proposal

There are no major modifications from the proposal that we had provided originally. All proposed AWS technologies mentioned in the proposal have been leveraged in the application.

Uniqueness

In order to create a unique URL for the same image name that people might upload at different times, we have retrieved the filename and append the timestamp to the name to achieve a unique name for each file. Simultaneously, we also create a record for that image in our MySQL based RDS database to track and retrieve the uploads.

Test Cases and Test Plan Execution

Test Case	Test Category	Execution Plan	Test Result
All users to have a unique name	Design	Try to signup a user with a username that already accepts, signup must fail	Pass
Login not allowed until account is verified	Design	If the account is not verified, user must not be able to view or upload images	Pass
Image Upload	Usability	Image must be uploaded correctly from the client's browser	Pass
S3 and RDS consistency for new image	Design	When a new image is uploaded, a new record must be created in RDS with the correct S3 link and all other image and user related details	Pass
Search images by user	Usability	Search for images uploaded by a specific user	Pass
Search images by keywords	Usability	Search for images upload for a specific keyword	Pass
Check if images are sorted by date	Usability	Upload multiple images and test whether they are sorted in descending order by upload time	Pass
Scale out: Addition of more instances when network traffic increases	Configuration	Upload multiple large images simultaneously as different users to check if the number of EC2 instances are automatically added	Pass
Scale in: Deletion of additional instances when network traffic decreases	Configuration	Minimize the network traffic after a high load to see if the additional EC2 instances are terminated	Pass

Project Postmortem

Issues uncovered:

1. One issue that we uncovered was to identify the linked entities between Cognito, S3, and RDS.

2. Another issue that we uncovered was to test if the auto scaling and load balancer were working correctly on the test environment.

Implement something differently:

1. The solution that we developed for the first issue was by using RDS as the intermediary between Cognito and S3. RDS links to Cognito users using the username (always unique due), and also links to S3 using the Image URL (unique due to timestamp).

2. The solution that we developed for testing auto scalers and load balancers is to set very low upper threshold for increasing the number of instances. We set the upper threshold to 1 mb “Network In” and uploaded images of about 15 mb within a minute to check if new EC2 instances are added.

Potential Improvements:

1. One potential improvement could be more structured tables in the data for more normalization of data.

2. Another potential improvement could be enhancing a more intuitive UI for the application.