INFO 7374 – Algorithmic Digital Marketing

Assignment 3- Implementing Visual Search

Anvesh Varma Vatsavai - 002190816

Vishnu Teja Pakalapati - 002197295

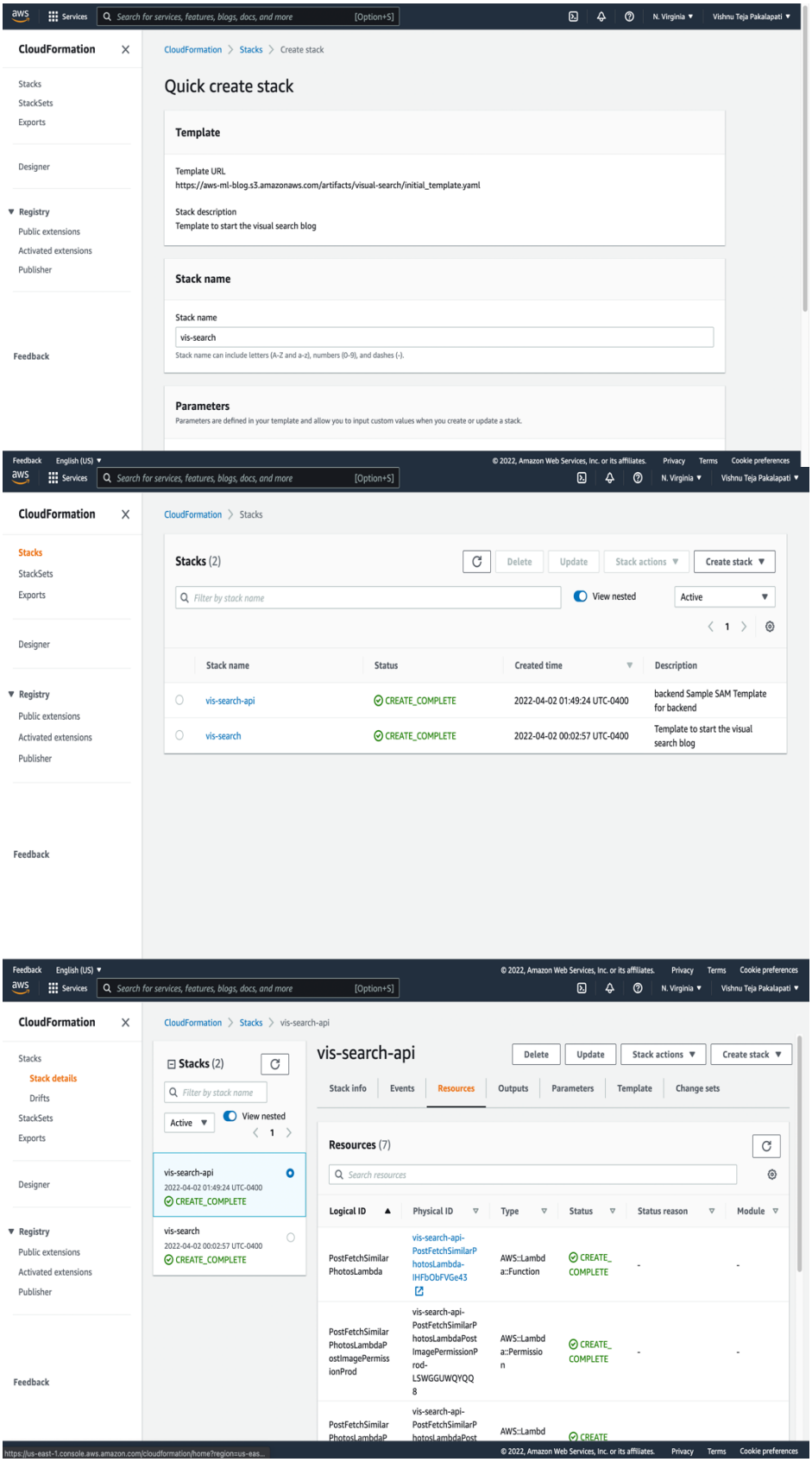
OBJECTIVE

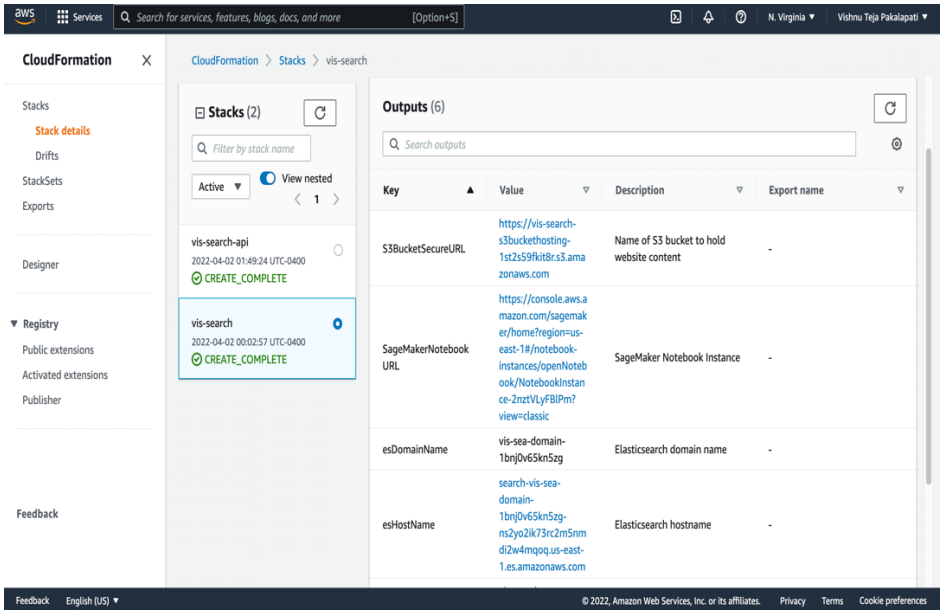
QU has hired us as an Algorithmic Marketing Analysts. QU is a consulting organization specializing in marketing analytical solutions

Our objective is to implement Visual search, to enhance the user experience of the customers by providing rich and engaging interfaces and to help ease their shopping experience

1. Building a visual search application with amazon sage maker and Implementing Amazon ES Pipeline
2. Finding similar images using ikatsaov/tensor house and integrating with Streamlit
3. Implementing elastic search service locally using python client and flask to in just a dataset from a file into elastic search so that each row in CSV file is turned into a document – reference elasticsearch py bulk injust

1.Building Amazon ES Pipeline using Amazon sage maker and Amazon API's


CloudFormation
Stacks
StackSets
Exports
Designer
Registry
Public extensions
Activated extensions
Publisher
Feedback
CloudFormation > Stacks > Create stack
Quick create stack
Template
Template URL
https://aws-ml-blog.s3.amazonaws.com/artifacts/visual-search/initial_template.yaml
Stack description
Template to start the visual search blog
Stack name
Stack name
vis-search
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
CloudFormation > Stacks
Stacks (2)
Delete
Update
Stack actions
Create stack
Filter by stack name
View nested
Active
Stack name
Status
Created time
Description
vis-search-api
CREATE_COMPLETE
2022-04-02 01:49:24 UTC-0400
backend Sample SAM Template for backend
vis-search
CREATE_COMPLETE
2022-04-02 00:02:57 UTC-0400
Template to start the visual search blog
CloudFormation > Stacks > vis-search-api
vis-search-api
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Resources (7)
Search resources
Logical ID
Physical ID
Type
Status
Status reason
Module
PostFetchSimilarPhotosLambda
vis-search-api-PostFetchSimilarPhotosLambda-IHFbObFVGe43
AWS::Lambda::Function
CREATE_COMPLETE
PostFetchSimilarPhotosLambdaPostImagePermissionProd
vis-search-api-PostFetchSimilarPhotosLambdaPostImagePermissionProd-LSWGGUWQYQQ8
AWS::Lambda::Permission
CREATE_COMPLETE



2. Finding similar images using ikatsaov/tenso

```
image_paths = glob.glob('/content/drive/MyDrive/ADM/*.jpg')
print(f'Founnd [{len(image_paths)}] images')

images = {}
for image_path in image_paths:
    image = cv2.imread(image_path, 3)
    b,g,r = cv2.split(image)          # get b, g, r

    image = cv2.merge([r,g,b])        # switch it to r, g, b
    image = cv2.resize(image, (200, 200))
    images[ntpath.basename(image_path)] = image

n_col = 8
n_row = int(len(images)/n_col)
f, ax = plt.subplots(n_row, n_col, figsize=(16, 8))
for i in range(n_row):
    for j in range(n_col):
        ax[i, j].imshow(list(images.values())[n_col*i + j])
        ax[i, j].set_axis_off()
```

```
Founnd [389] images
```

r house

```
image_style_embeddings = {}
[4] for image_path in tqdm(image_paths):
        image_tensor = load_image(image_path)
        style = style_to_vec( image_to_style(image_tensor) )
        image_style_embeddings[ntpath.basename(image_path)] = style
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_or
80142336/80134624 [==============================] - 1s 0us/step
80150528/80134624 [==============================] - 1s 0us/step
100%|          | 389/389 [12:32<00:00,  1.93s/it]
```

```
def search_by_style(reference_image, max_results=10):
    v0 = image_style_embeddings[reference_image]
    distances = {}
    for k,v in image_style_embeddings.items():
        d = sc.spatial.distance.cosine(v0, v)
        distances[k] = d

    sorted_neighbors = sorted(distances.items(), key=lambda x: x[1], reverse=False)

    f, ax = plt.subplots(1, max_results, figsize=(16, 8))
    for i, img in enumerate(sorted_neighbors[:max_results]):
        ax[i].imshow(images[img[0]])
        ax[i].set_axis_off()

    plt.show()
```

```
[8] # images mostly match the reference style, although not perfectly
    search_by_style('img_00000143.jpg')
```

3.Integarting similarity search with streamlit

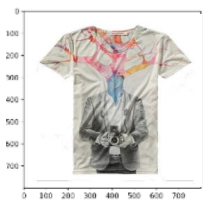
Visual Search

Search 10 Similar Images

Upload file

H3d7fb2977f0b4b099f7df239bba1fadda.jpg

browse files



STREAMLIT Output :