Web scraping

Overview

This script allows users to download an image from a given URL and save it to their local system. The script is implemented in Python using the `requests` library to handle HTTP requests and basic file operations to save the image locally.

Requirements

To use this script, you need to have Python installed along with the `requests` library.

Usage

1. Run the code :

import requests

# Function to download an image from a URL

def download\_image(image\_url, file\_name="downloaded\_image.jpg"):

# Send a GET request to the URL

response = requests.get(image\_url

# Check if the request was successful

if response.status\_code == 200:

# Write the content of the response to a file

with open(file\_name, 'wb') as file:

file.write(response.content)

print(f"Image successfully downloaded as {file\_name}")

else:

print("Failed to download image. Status code:", response.status\_code)

# Provide the image URL

image\_url = input("Enter the image URL ")

# Call the function to download the image

download\_image(image\_url)

1. Enter the image URL:

When prompted, enter the full URL of the image you wish to download.

Example

If you enter an image URL like `https://example.com/sample.jpg`, the image will be downloaded and saved as `downloaded\_image.jpg` in the current directory.

fetch data from amazon AWS

## Overview

This Python script is designed to connect to an AWS S3 bucket, list the contents of a specified folder, and download the files to a local directory. It uses the `boto3` library for interacting with AWS services and standard Python libraries for file and directory operations.

Requirements

- Python 3.x

- `boto3` library for AWS service interactions

- Proper AWS credentials with access permissions for the S3 bucket

AWS Credentials

Ensure that your AWS credentials are properly configured for secure access:

- Environment Variables: Set `AWS\_ACCESS\_KEY\_ID` and `AWS\_SECRET\_ACCESS\_KEY`.

- AWS Credentials File: Store credentials in `~/.aws/credentials`.

Usage

1.Configure Download Directory: Update the `download\_dir` variable to specify your preferred download location.

3. Run the Script

```

python s3\_file\_downloader.py

```

Explanation of the Code

### Key Components

- `boto3.client()`: Initializes the client for connecting to AWS S3 using specified credentials and region.

- `list\_objects\_v2()`: Lists objects within the given bucket and folder prefix.

- `download\_file()`: Downloads each file from the S3 bucket to the specified local directory.

- `os.makedirs()`: Creates the local download directory if it doesn't exist.

- `os.chmod()`: Ensures the directory has write permissions.

### Functionality

- `fetch\_from\_s3()`:

- Checks if the local download directory exists and creates it if not.

- Connects to the AWS S3 bucket using the `boto3` client.

- Lists and downloads files from the `dataset/` folder in the specified S3 bucket.

- `fetch\_data()`:

- Main function that calls `fetch\_from\_s3()` and indicates when the operation is complete.

## Security Best Practices

- Avoid Hard-Coding Credentials: Do not hard-code your AWS `access\_key\_id` and `secret\_access\_key` in the code. Use environment variables, AWS credential profiles, or IAM roles for secure management.

- Set Proper Permissions: Ensure the user or role has the required permissions (e.g., `s3:ListBucket` and `s3:GetObject`) to access the S3 bucket.

Example Output

Fetching data from AWS S3 bucket...

Downloading example\_file.jpg...

example\_file.jpg downloaded successfully to C:\Users\DELLL\Downloads\aws\_download.

...

Data fetch complete.

```