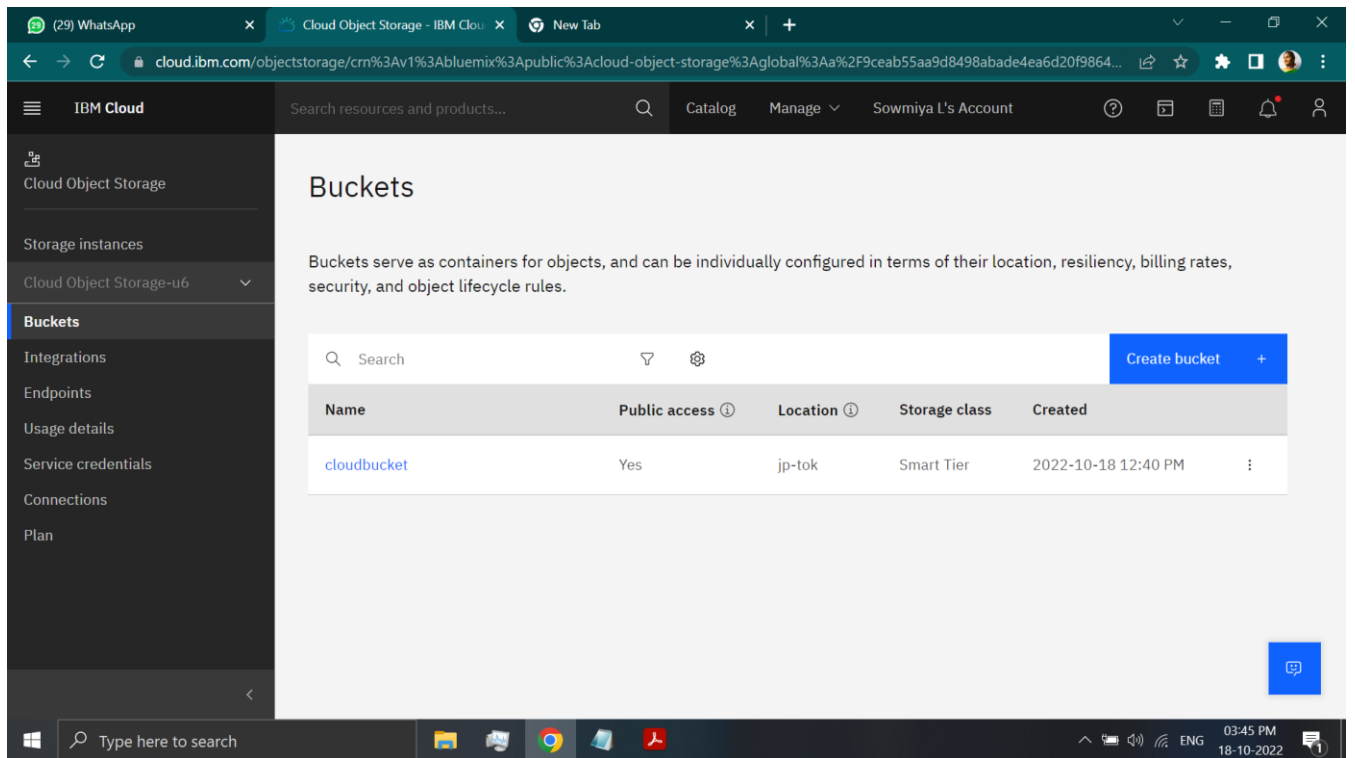


Assignment-3

Date	10 October 2022
Team ID	PNT2022TMID30593
Project Name	Smart Fashion Recommender Application

1. CREATE A BUCKET IN IBM OBJECT STORAGE.



2.Upload an 5 images to ibm object storage and make it public. write html code todisplaying all the 5 images.

IBM Cloud

Search resources and products...

Cloud Object Storage

Storage instances

Cloud Object Storage-u6

Buckets

Integrations

Endpoints

Usage details

Service credentials

Connections

Plan

cloudbucket

Transfers Details Actions...

Objects Configuration Permissions

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads [Learn more](#)

Prefix filter

Upload

Object name	Archived ⓘ	Size	Last modified
<input type="checkbox"/> download (1).jpg		7.3 KB	2022-10-18 2:55 PM
<input type="checkbox"/> download.jpg		5.6 KB	2022-10-18 2:55 PM
<input type="checkbox"/> images (1).jpg		6.8 KB	2022-10-18 2:55 PM
<input type="checkbox"/> images (2).jpg		9.3 KB	2022-10-18 2:55 PM
<input type="checkbox"/> images.jpg		4.6 KB	2022-10-18 2:55 PM

Type here to search

03:03 PM 18-10-2022

IBM Cloud

Search resources and products...

Cloud Object Storage

Storage instances

Cloud Object Storage-u6

Buckets

Integrations

Endpoints

Usage details

Service credentials

Connections

Plan

Manage access to this bucket by creating IAM policies for users and service IDs. Users and service IDs must also have an instance level viewer role (or higher) to use the console or to list buckets using the REST API.

Access policies

Public access

Access policy update

Access group policy created

A new access policy for this bucket was created for the group:
Public Access
To delete/edit go to the [IAM console](#).

Role for this bucket:

Content Reader

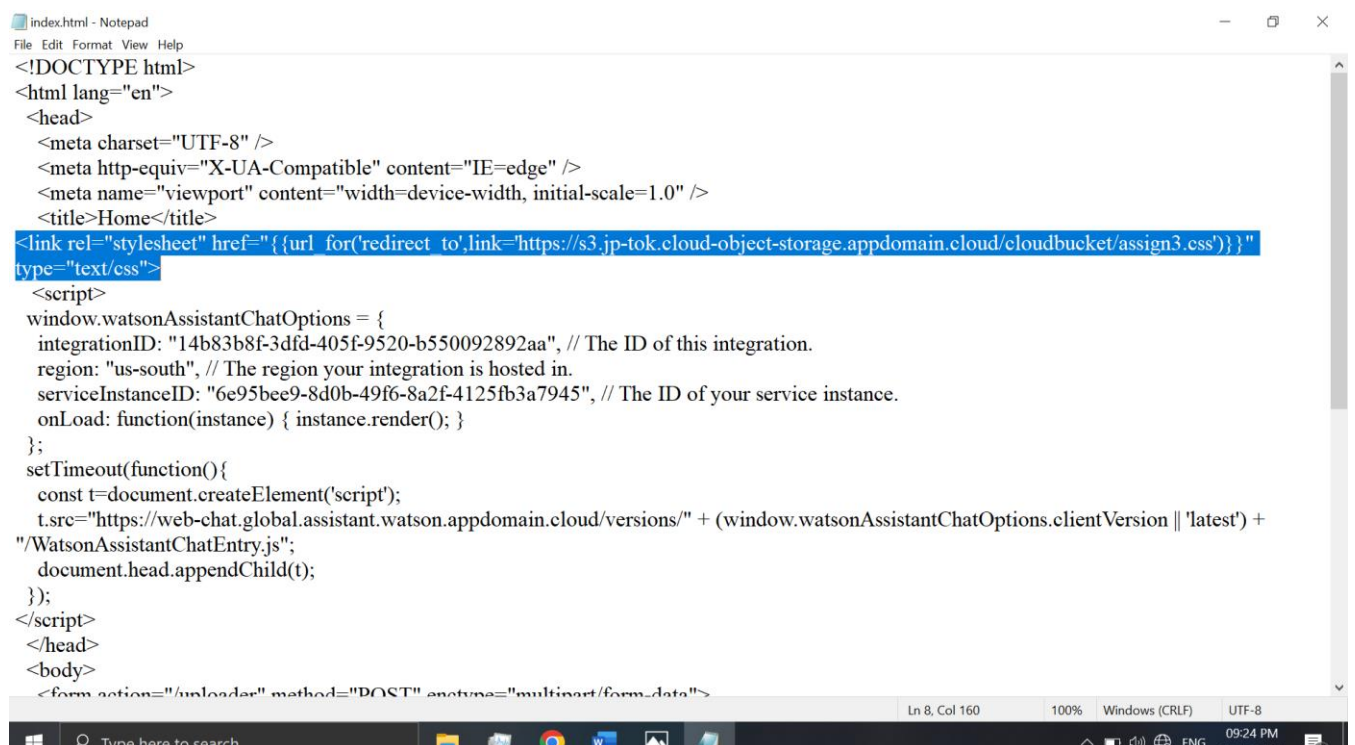
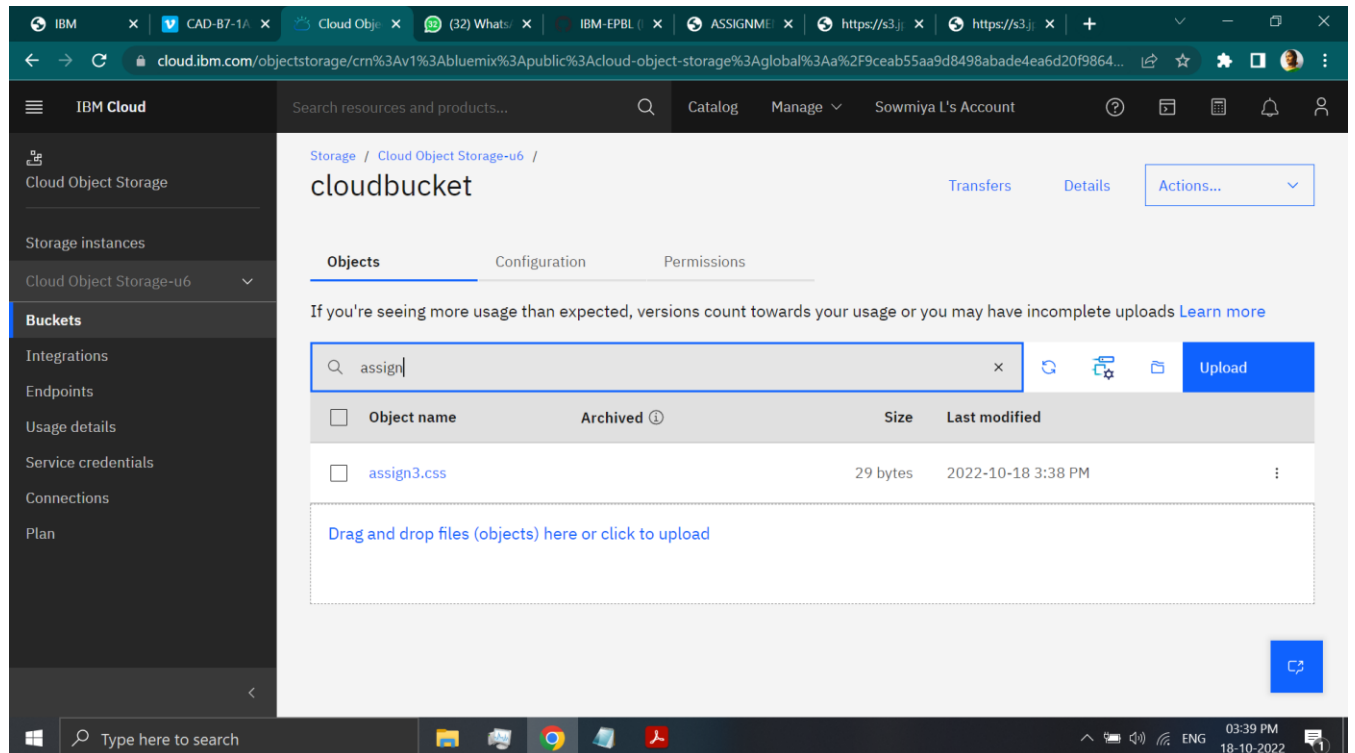
As a Content Reader, one can read and list objects in the bucket.

Create access policy

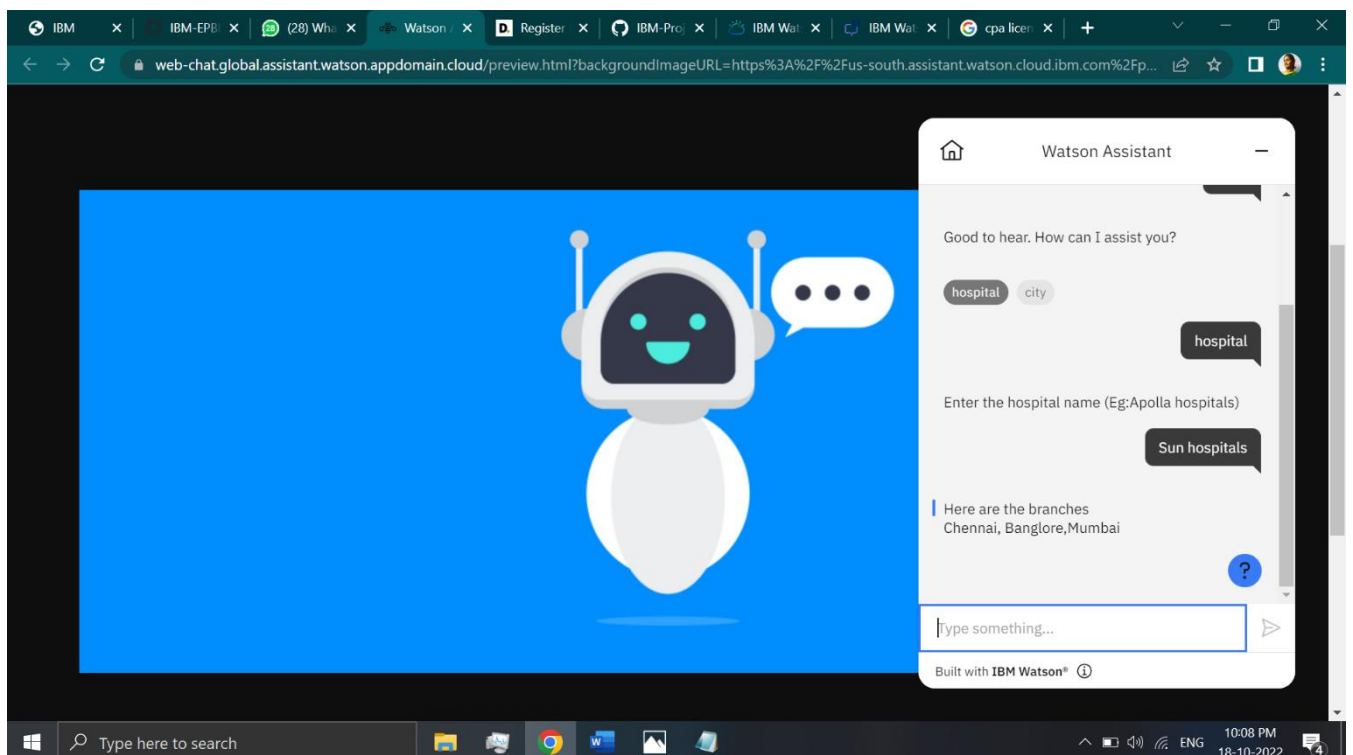
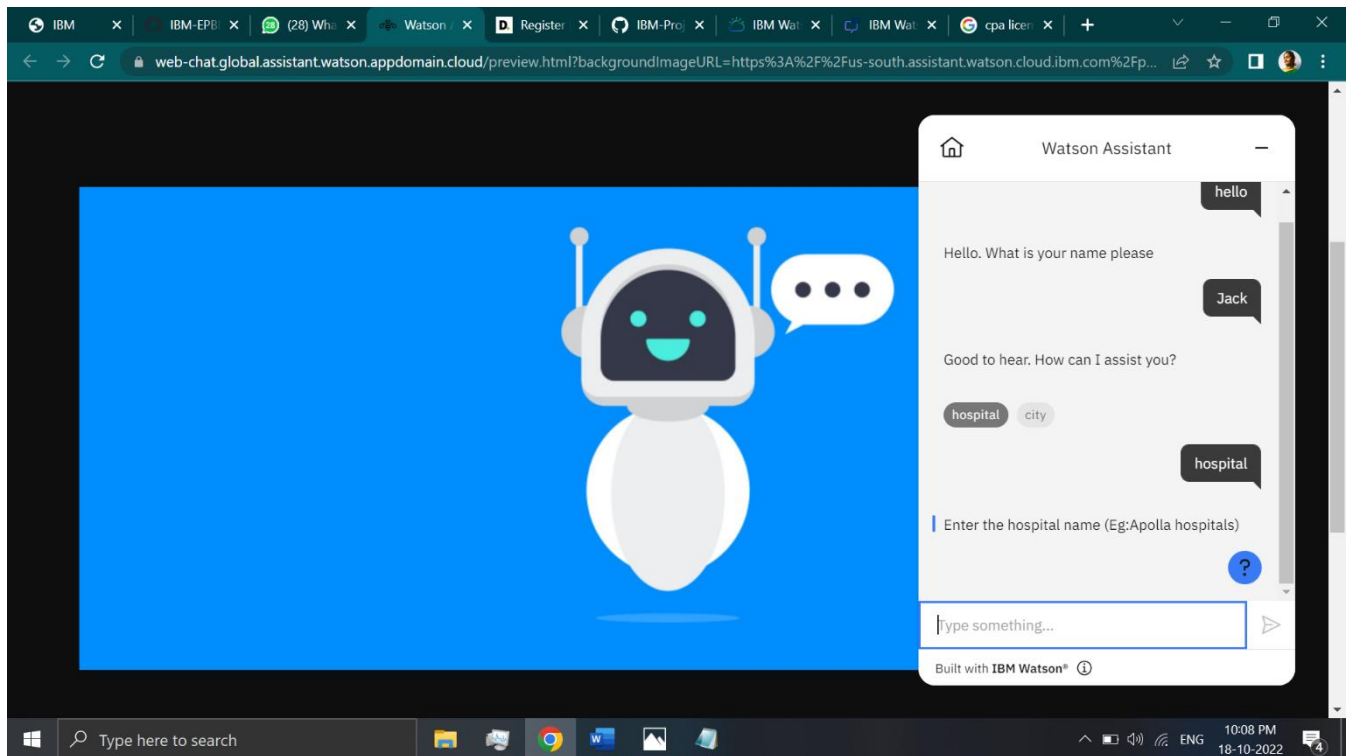
Type here to search

12:48 PM 18-10-2022

2. Upload a css page to the object storage and use the same page in your HTML code.



3.Design a chatbot using IBM Watson assistant for hospital.



Web URL for Assistant:

<https://web-chat.global.assistant.watson.appdomain.cloud/preview.html?backgroundImageURL=https%3A%2F%2Fus-south.assistant.watson.cloud.ibm.com%2Fpublic%2Fimages%2Fupx-6e95bee9-8d0b-49f6-8a2f-4125fb3a7945%3A%3Ab437fcda-6135-46e5-9e62-faa901cdce2d&integrationID=14b83b8f-3dfd-405f-9520-b550092892aa®ion=us-south&serviceInstanceID=6e95bee9-8d0b-49f6-8a2f-4125fb3a7945>

4. Create Watson assistant service with 10 steps and use 3 conditions in it. Load thatscript in HTML page.

The screenshot displays the IBM Watson Assistant configuration interface. On the left, a chat window shows a conversation with 10 steps. The steps are:

- Step 5: Appointment details. You can make appointment by contacting this number 04257xxxxx. (Re-ask previous step(s))
- Step 8: Parking facility. Yes, this hospital has parking facility. (Re-ask previous step(s))
- Step 9: Bed Availability. Today 50 beds available. Thank you. (Action complete)
- Step 10: (Empty step)

On the right, the configuration panel shows the assistant's response: "Today 50 beds available. Thank you". Below this, there is a section for "And then" with the action "End the action".

The bottom status bar shows the temperature as 29°C Cloudy, the time as 11:34, and the date as 19-10-2022.

Figure 1. 10 steps of conversation

Included 3 conditions in steps:

The screenshot shows the IBM Watson Assistant interface. On the left, a conversation flow diagram highlights Step 3, which is a text input step with the prompt "Enter the hospital name (Eg: Apollo hospitals)". The step is configured with a condition: "2 == hospital . city". The right panel shows the configuration for Step 3, which is taken "with conditions". The conditions are defined as "If Any of this is true:" followed by two conditions: "2. Good to he... is hospital" and "2. Good to he... is city". The "Assistant says" section shows the prompt "Enter the hospital name (Eg: Apollo hospitals)".

The screenshot shows the IBM Watson Assistant interface. On the left, a conversation flow diagram highlights Step 5, which is a text input step with the prompt "Hospital is 30km away from your location". The step is configured with a condition: "4 == Distance". The right panel shows the configuration for Step 5, which is taken "with conditions". The conditions are defined as "If All of this is true:" followed by one condition: "4. Here are th... is Distance". The "Assistant says" section shows the prompt "Hospital is 30km away from your location".

IBMYour Platform(1) WhatsAppProject PlanIBMResource listIBM WatsonIBM Watson

us-south.assistant.watson.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Aconversation%3Aus-south%3Aa%2F9ceab55aa9d8498abade4ea6d20f9864%3A6e9...

GmailYouTubeMapsNew TabHomeA Tour of Google Cl...

IBM Watson Assistant LiteUpgradeFashion BotLearning center?

Hello

4 == Appointment details

7

You can make appointment by contacting this number 04257xxxxx

Continue to next step

4 == Parking facility

8

Yes, this hospital has parking facility

Continue to next step

Thank you.

9

Continue to next step

This step has no content

10

New step +

Step 8 is taken with conditions

f_x

Conditions1 condition

If All of this is true:

4. Here are th... is Parking facility

and Add condition +

New condition group +

Assistant says

B I

Yes, this hospital has parking facility

Preview

29°C Cloudy

ENG IN11:1619-10-2022

Index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Home</title>

    <link rel="stylesheet" href="{{ url_for('redirect_to',link='https://s3.jp-tok.cloud-object-storage.appdomain.cloud/cloudbucket/assign3.css')}} " type="text/css">

    <script>

      window.watsonAssistantChatOptions = {

        integrationID: "14b83b8f-3dfd-405f-9520-b550092892aa", // The ID of this integration.

        region: "us-south", // The region your integration is hosted in.

        serviceInstanceID: "6e95bee9-8d0b-49f6-8a2f-4125fb3a7945", // The ID of your service instance.

        onLoad: function(instance) { instance.render(); }

      };

      setTimeout(function(){

        const t=document.createElement('script');

        t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";

        document.head.appendChild(t);

      });

    </script> </head>

    <body>

      <form action="/uploader" method="POST" enctype="multipart/form-data">

        <input type="text" placeholder="Enter file name" name="filename" />

        <br />

        <br />

        <input type="file" name="file" />

        <br />

        <br />

        <input type="submit" />

      </form>

    </body>

  </html>
```



```

</form>

<br/>

<br/>

<br/>

{% for row in files %}

    <div style="border: 1px solid #EFEFEF;margin:10px;">

        <h3>Filename : {{ row }} </h3>

        </td>

    </div>

{% endfor %}

</body>

</html>

```

App.py

```

import io

from flask import Flask,redirect,url_for,render_template,request

import ibm_boto3

from ibm_botocore.client import Config, ClientError

COS_ENDPOINT="https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS_API_KEY_ID=""

COS_INSTANCE_CRN=""


cos = ibm_boto3.resource("s3",

    ibm_api_key_id=COS_API_KEY_ID,

    ibm_service_instance_id=COS_INSTANCE_CRN,

    config=Config(signature_version="oauth"),

    endpoint_url=COS_ENDPOINT

)

```

```

app=Flask(__name__)

@app.route('/')
def index():
    try:
        files = cos.Bucket('cloudbucket').objects.all()
        files_names = []
        for file in files:
            files_names.append(file.key)
            print(file)
            print("Item: {0} ({1} bytes)".format(file.key, file.size))
        return render_template('index.html',files=files_names)

    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
        return render_template('index.html')
    except Exception as e:
        print("Unable to retrieve bucket contents: {0}".format(e))
        return render_template('index.html')

@app.route('/uploader',methods=['POST'])
def upload():
    name_file=request.form['filename']
    f = request.files['file']
    try:
        part_size = 1024 * 1024 * 5

        file_threshold = 1024 * 1024 * 15

        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,

```

```
        multipart_chunksize=part_size
    )
```

```
content = f.read()
cos.Object('cloudbucket', name_file).upload_fileobj(
    Fileobj=io.BytesIO(content),
    Config=transfer_config
)
return redirect(url_for('index'))
```

```
except ClientError as be:
    print("CLIENT ERROR: {0}\n".format(be))
    return redirect(url_for('index'))
```

```
except Exception as e:
    print("Unable to complete multi-part upload: {0}".format(e))
    return redirect(url_for('index'))
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```