# Media Streaming With IBM Cloud Video Streaming Phase 2-Development Part-2

Streaming media with IBM Cloud Video Streaming typically involves using their API and services. Here's a high-level step-by-step guide in Python using the IBM Cloud Video Streaming API:

#### **1.** Get IBM Cloud Account:

- Sign up for an IBM Cloud account if you don't have one.
- Set up a new IBM Cloud Video Streaming instance.

### 2. Get API Key:

- Generate an API key for your IBM Cloud account.

### 3. Install Required Libraries:

- Install the necessary Python libraries using pip:

```bash

Pip install requests

...

## 4. Write Python Code:

- Use the following Python code as a starting point.

Replace `<YOUR\_API\_KEY>` and `<STREAM\_ID>` with your actual API key and stream ID.

```
```python
  Import requests
  Api key = "<YOUR API KEY>"
 Stream_id = "<STREAM_ID>"
 Base_url = https://api.video.ibm.com
  # Create a live channelDef
  create channel():
    url = f"{base url}/channels"
    headers = {"Content-Type": "application/json", "Authorization":
f"Bearer {api key}"}
    payload = {"name": "MyLiveChannel", "broadcasting":
True}
    response = requests.post(url, json=payload,
headers=headers)
    return response.json()["id"]
  # Get the RTMP URL for the channelDef
  get_rtmp_url(channel_id):
    url = f"{base url}/channels/{channel id}/ingest"
```

```
headers = {"Content-Type": "application/json", "Authorization":
f"Bearer {api key}"}
    response = requests.get(url, headers=headers)return
response.json()["streaming"]["ingestPoints"][0]["url"]
  # Start streaming
 Def start_streaming(channel_id):
    url = f"{base url}/channels/{channel id}/broadcast"headers =
    {"Content-Type": "application/json",
"Authorization": f"Bearer {api key}"}
    payload = {"broadcasting": True}
    response = requests.patch(url, json=payload,headers=headers)
    return response.status_code == 200
 if ____ == "___ main____":
    # Create a channel
    Channel id = create channel()
    # Get RTMP URL
    Rtmp_url = get_rtmp_url(channel_id)
```

```
# Start streaming
Success = start_streaming(channel_id)
    If success:
        Print(f"Stream started successfully. RTMP URL:
{rtmp_url}")
    Else:
        Print("Failed to start streaming.")
```

## 5. Run the Code:

- Save the code in a file (e.g., `streaming\_example.py`) and run it. This example code creates a live channel, retrieves the RTMP URL for the channel, and starts streaming. Make sure to handle errors appropriately and customize the codebased on your specific requirements.

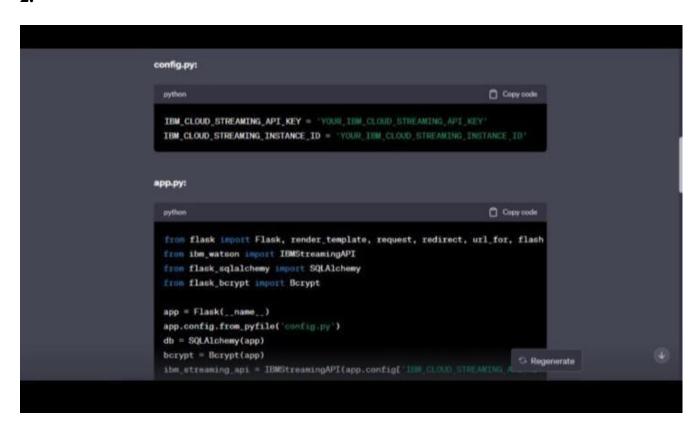
#### **IMAGES:**

These images are some sample images.

#### 1. File structure:



2.



```
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_sqlalchemy import IBMStreamingAPI
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import SQLAlchemy
from flask_bcrypt import SQLAlchemy

app = Flask(__name__)
app.config.from_pyfile('config.py')
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
ibm_streaming_api = IBMStreamingAPI(app.config['IBM_CLGUD_STREAMING_API_KEY']

# Define your User model here (id, username, email, password fields)

@app.route('/')
dof index():

# Add legic to retrieve and display movies from IBM Cloud Streaming
return render_template('index.html')

@app.route('/login', esthods=f'GET', 'POST'))
```

```
@app.route('/')
def index():
    # Add logic to retrieve and display movies from IBM Cloud Streaming
    return render_template('index.html')

@app.route('/legin', methods=['GET', 'POST'])
def login():
    # Add logic logic here (check username and password, validate user)
    return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    # Add user registration logic here (create user, store user data)
    return render_template('register.html')

if __name__ == '__main__':
    app.run(debug=True)
```

5.

7.

```
html Copycode

<IDOCTYPE html>
<html lang="en">
<html lan
```