

Media Stream with IBM cloud video streaming

PHASE 4: 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 __name__ == "__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.")
'''
```

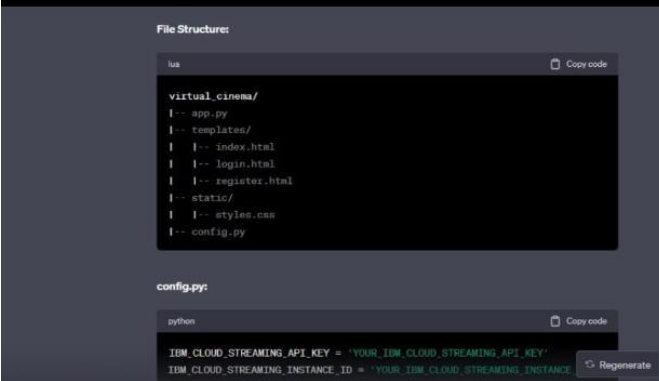
## 5. Run the Code:

- 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 code based on your specific requirements.

## IMAGES:

These images are some sample images .

## 1. File structure:

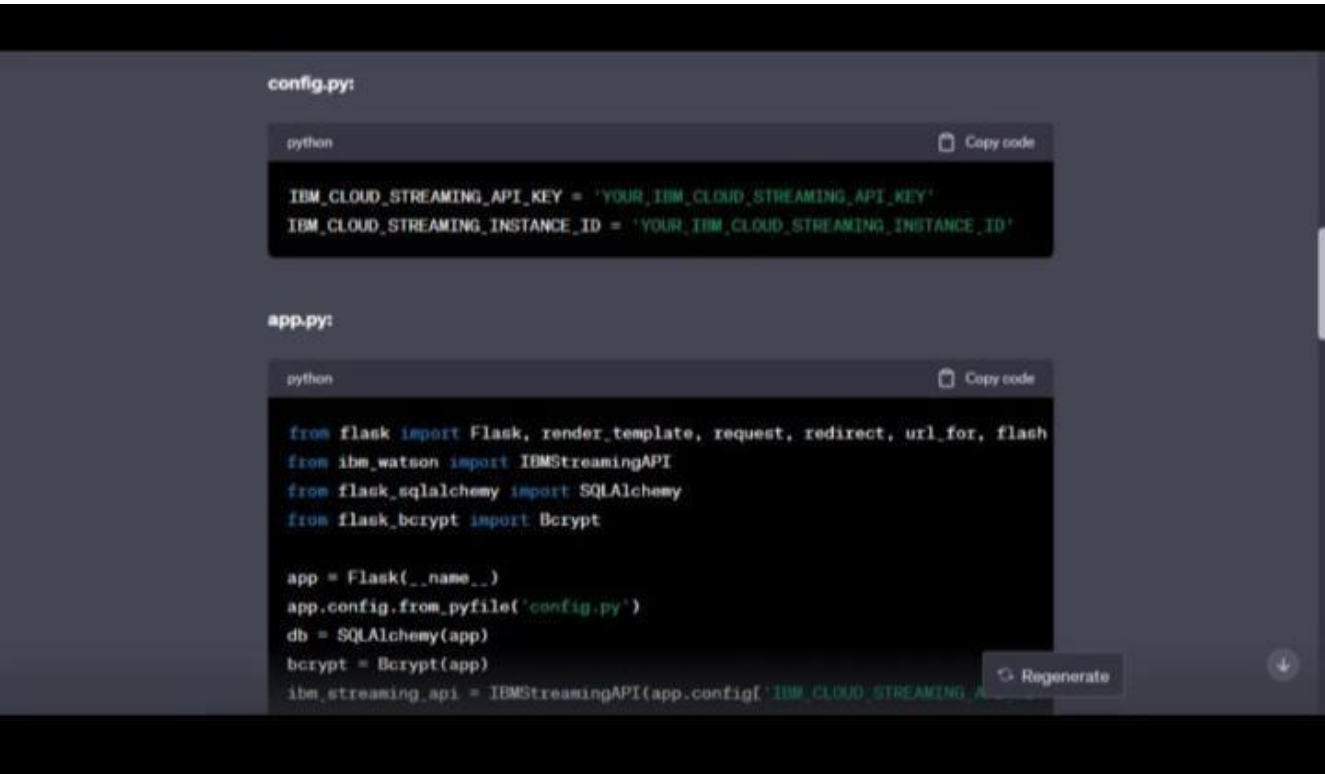


The screenshot shows a 'File Structure' panel with a tree view of a project. The root is 'lua', which contains a subdirectory 'virtual\_cinema/'. Inside 'virtual\_cinema/', there are files 'app.py', 'templates/', 'index.html', 'login.html', 'register.html', 'static/', 'styles.css', and 'config.py'. Below the tree, the 'config.py' file is expanded, showing its contents in a code editor. The code defines two variables: 'IBM\_CLOUD\_STREAMING\_API\_KEY' and 'IBM\_CLOUD\_STREAMING\_INSTANCE\_ID', both set to placeholder values. There are 'Copy code' and 'Regenerate' buttons next to the code editor.

```
lua
└-- virtual_cinema/
 |-- app.py
 |-- templates/
 | |-- index.html
 | |-- login.html
 | |-- register.html
 |-- static/
 |-- styles.css
 |-- config.py

config.py:
python
IBM_CLOUD_STREAMING_API_KEY = 'YOUR_IBM_CLOUD_STREAMING_API_KEY'
IBM_CLOUD_STREAMING_INSTANCE_ID = 'YOUR_IBM_CLOUD_STREAMING_INSTANCE_ID'
```

2.

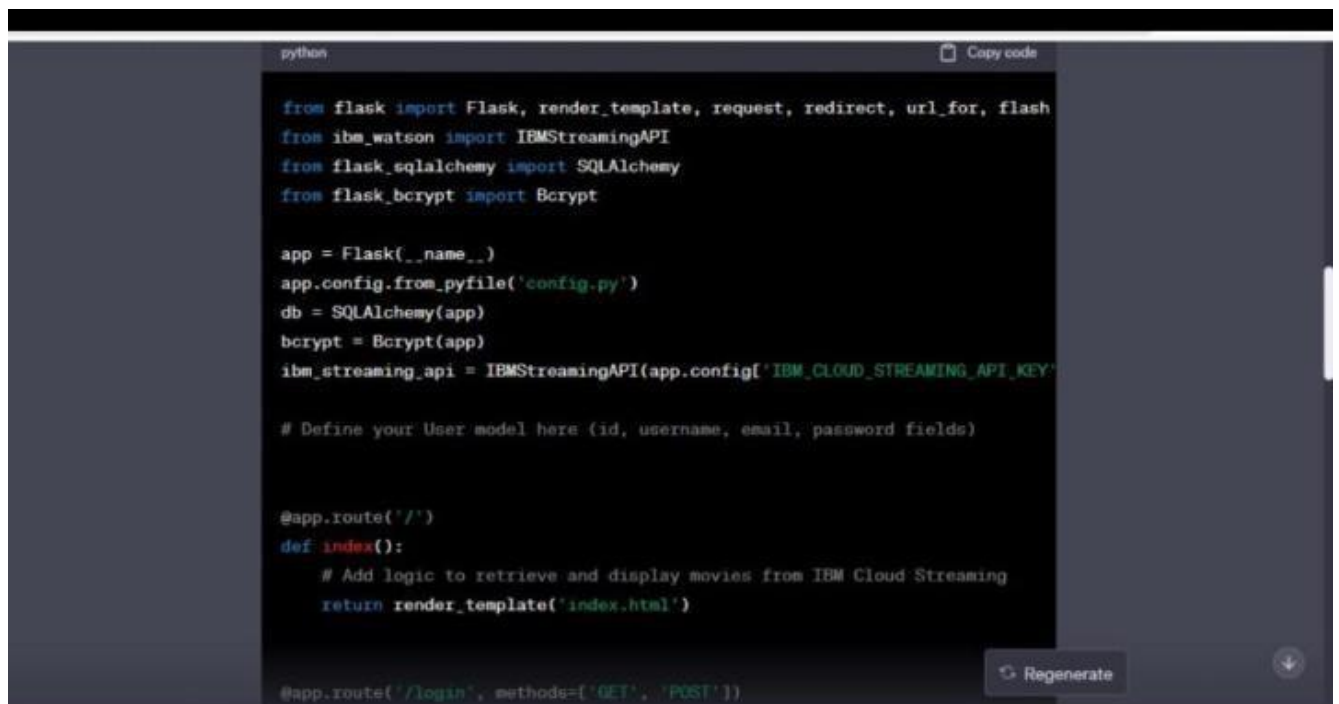


The screenshot shows two code editors. The top editor is for 'config.py' and contains the same configuration code as seen in the previous image. The bottom editor is for 'app.py' and contains the following code:

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

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

3.



The image shows a code editor window with a dark theme. The code is written in Python and defines a Flask application. It includes imports for Flask, IBM Streaming API, SQLAlchemy, and Bcrypt. The application is configured with a config file and a database. A route is defined for the index page, which uses the IBM Streaming API to retrieve movies. The code is as follows:

```
python Copy code

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

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

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

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

@app.route('/login', methods=['GET', 'POST'])
```

At the bottom right of the editor, there is a "Regenerate" button and a download icon.

4.

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

@app.route('/login', methods=['GET', 'POST'])
def login():
 # Add login 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)
```

Regenerate

5.

register.html

```
html Copy code
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
 <title>Register</title>
</head>
<body>
 <!-- Registration form goes here -->
</body>
</html>
```

styles.css: (Example styles, modify as per your requirements)

Regenerate



6.

styles.css: (Example styles, modify as per your requirements)

```
css Copy code

body {
 font-family: Arial, sans-serif;
 margin: 0;
 padding: 0;
 background-color: #f4f4f4;
}

.container {
 width: 80%;
 margin: auto;
 overflow: hidden;
}

/* Add your styles for login and registration forms here */
```

7.

index.html:

```
html Copy code

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <link rel="stylesheet" href="{ url_for('static', filename='styles.css') }">
 <title>Virtual Cinema</title>
</head>
<body>
 <!-- Display movies and video player here -->
</body>
</html>
```

8.

login.html:

```
html Copy code

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
 <title>Login</title>
</head>
<body>
 <!-- Login form goes here -->
</body>
</html>
```