**实现流程**

**用户上传视频：**

**用户通过前端界面选择视频文件。**

**前端通过 JavaScript 获取用户的网络信息（如带宽、网络类型），并发送给后端。**

**生成 S3 预签名上传 URL：**

**前端将文件名（或其它必要信息）发送到 AWS Lambda 的 API。**

**Lambda 函数生成一个 S3 预签名上传 URL，并将该 URL 返回给前端。**

**视频上传到 S3：**

**前端使用返回的 S3 预签名 URL，通过 PUT 请求上传视频文件到 S3。**

**上传完成后的处理：**

**Optional：视频上传成功后，前端可以显示成功消息（如“上传完成”）。**

**上传完成触发视频处理： 视频上传到 S3 后，触发 EC2 实例 开始处理视频。 EC2 实例使用 FFmpeg 将视频转码成不同的分辨率，选择一个适合用户网络的分辨率（例如 720p、480p）。 处理后的视频存储和下载： 处理完成后，视频上传回 S3。 前端提供下载链接，用户可以下载处理后的视频**

**第一天初步工作**

**1. 前端工作：**

**创建视频上传界面：设计一个简单的用户界面，允许用户选择视频文件（<input type="file">）。**

**获取网络信息：使用 JavaScript 获取用户的网络信息（如带宽、网络类型），并通过 fetch 或 XMLHttpRequest 发送给后端。这一步可以是简化版，不需要精确的网络优化，只需要把网络信息传给 Lambda 即可。**

**发送文件名到后端：前端将文件名或其它必要信息发送到 AWS Lambda API 端点（通过 HTTP POST 请求）。**

**上传文件到 S3：前端接收到 Lambda 返回的 S3 预签名上传 URL 后，通过 fetch 或 XMLHttpRequest 将视频文件上传到 S3。**

**前端代码实现示例：**

**javascript**

**// 选择文件并发送文件名到 Lambda API 生成预签名 URLasync function getUploadUrl(fileName) {**

const response = await fetch('https://your-api-id.execute-api.region.amazonaws.com/your-resource', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ file\_name: fileName })

});

const data = await response.json();

return data.upload\_url; // 返回 S3 上传 URL

}

// 使用预签名 URL 上传文件到 S3async function uploadToS3(uploadUrl, file) {

const response = await fetch(uploadUrl, {

method: 'PUT',

body: file

});

if (response.ok) {

console.log('Upload success!');

} else {

console.error('Upload failed');

}

}

// 上传视频的流程const fileInput = document.getElementById('fileInput');

fileInput.addEventListener('change', async (event) => {

const file = event.target.files[0];

if (file) {

const uploadUrl = await getUploadUrl(file.name); // 获取上传 URL

await uploadToS3(uploadUrl, file); // 上传文件到 S3

}

});

**2. 后端工作：**

**1.创建 AWS Lambda 函数：创建一个 Lambda 函数，接收前端传来的文件名（或其它必要信息），并生成一个 S3 预签名上传 URL 返回给前端。**

* S3 预签名上传 URL（S3 pre-signed upload URL）是一个整体的概念，通常指的是由 AWS 生成的一个 URL，这个 URL 包含了上传文件到 S3 的所有必要信息（包括权限和有效期），并且通常是用于 上传文件。
* “预签名”（pre-signed）表示这个 URL 是由 AWS 的授权机制生成的，授权允许你将文件上传到指定的 S3 存储桶，而无需暴露你的 AWS 密钥。
* S3：指的是 Amazon Simple Storage Service，是 AWS 提供的对象存储服务。

Lambda 代码实现示例：

python

import jsonimport boto3from botocore.exceptions import ClientErrorfrom datetime import datetime, timedelta

s3\_client = boto3.client('s3')

def lambda\_handler(event, context):

bucket\_name = "your-s3-bucket-name" # 替换为你的 S3 存储桶名称

file\_name = event['file\_name'] # 从前端传来的文件名

expiration = 3600 # 预签名 URL 有效期（单位：秒）

# 生成 S3 预签名的上传 URL

try:

upload\_url = s3\_client.generate\_presigned\_url('put\_object',

Params={'Bucket': bucket\_name, 'Key': file\_name},

ExpiresIn=expiration)

return {

'statusCode': 200,

'body': json.dumps({'upload\_url': upload\_url})

}

except ClientError as e:

return {

'statusCode': 500,

'body': json.dumps({'error': str(e)})

}

**2.配置 API Gateway：创建一个简单的 HTTP API 端点，在 AWS API Gateway 中配置，允许前端调用 Lambda 函数并返回预签名 URL。**

**配置 HTTP 方法（POST），并将 Lambda 函数绑定到该方法上。**

**（API Gateway 设置：创建一个新的 API。添加一个 POST 方法，配置为调用 Lambda 函数。配置跨域（CORS），以便前端可以发起跨域请求。）**

#### 3. ****测试与调试（之后做）****

* **测试 Lambda**：
  + 在 Lambda 控制台测试生成 S3 预签名 URL 的功能，确保返回 URL 可以在前端使用。
* **测试 API Gateway**：
  + 通过浏览器或 Postman 测试 API Gateway 是否能够正确调用 Lambda，并返回正确的 S3 预签名 URL。
* **测试前端上传**：
  + 前端与后端交互：确保能够成功获取预签名 URL 并将文件上传到 S3。

**主要的挑战可能来自 API Gateway 的 CORS 设置、Lambda 权限配置、文件大小及超时处理**