MultipartUpload

В прошлом уроке мы рассмотрели, как скачивать и загружать файлы без их полного сохранения в оперативной памяти.

Такая загрузка происходит в рамках одного соединения. Если в какой-то момент загрузка прервется, то нужно будет передавать весь файл заново.

Этот недостаток нестрашен, если файлы сравнительно маленького размера — 100-1000 Мб. Но когда нужно отправить файл в десятки или сотни гигабайт, могут возникнуть проблемы. Такими файлами могут быть дампы базы данных или логи сервиса.

Такие файлы лучше загружать частями, а потом соединять на стороне хранилища.

Для загрузки данных частями в протоколе S3 есть 4 специальных метода:

- CreateMultipartUpload — инициализирует загрузку частями
- UploadPart — загружает часть
- CompleteMultipartUpload — завершает загрузку
- AbortMultipartUpload — отменяет загрузку

Алгоритм загрузки

1. Вызываем **CreateMultipartUpload** и передаем в аргументы Bucket и Key. Из ответа нужно получить UploadId — это идентификатор загрузки. Он понадобится для однозначного определения, к какому файлу относится часть
2. В цикле вызываем **UploadPart**. В аргументы нужно передать Bucket, Key, PartNumber, UploadId, Body. PartNumber — номер части в последовательности (1, 2, 3, ...), UploadId — id из предыдущего метода. Из ответа нужно получить ETag — по сути, это версия. Если вызвать 2 раза метод UploadPart, то ETag позволит определить, какие данные нужно взять. Так нужно продолжать, пока все части не будут загружены
3. После загрузки всех частей нужна процедура завершения загрузки (объединение всех частей в один файл). Для этого вызываем **CompleteMultipartUpload** и передаем в аргументы UploadId и массив из двух ключей {"ETag": "", "PartNumber": 1}. Фунция может вызывать исключения:

- EntityTooSmall — часть меньше 5 Мб
- InvalidPart — какая-то часть не найдена
- InvalidPartOrder — порядок PartNumber не является возрастающим
- NoSuchUpload — UploadId не найдет

В случае, если произошла ошибка при загрузке, можно отменить загрузку методом **AbortMultipartUpload**. Для этого вызываем этот метод с аргументами Bucket, Key, UploadId.

Контекстный менеджер для алгоритма

```python
from typing import List, Optional


class MultipartUploader:
    def __init__(self, client, bucket: str, key: str) -> None:
        self.client = client
        self.bucket = bucket
        self.key = key

        self.part_number: int = 0
        self.parts: List[dict] = []
        self.mpu: Optional[dict] = None
        self.uploaded_size: float = 0
        self.is_loading: bool = False

    async def __aenter__(self):
        await self._create_uploading()
        return self

    async def __aexit__(self, exc_type, exc_val, exc_tb):
        if not exc_type:
            await self._finish_uploading()
        else:
            await self.client.abort_multipart_upload(
                Bucket=self.bucket,
                Key=self.key,
                UploadId=self.mpu["UploadId"],
            )

    async def _create_uploading(self) -> None:
        self.parts = []
        self.part_number = 1

        self.mpu = await self.client.create_multipart_upload(
            Bucket=self.bucket,
            Key=self.key,
        )

        self.is_loading = True
        self.uploaded_size = 0

    async def upload_part(self, chunk: bytes) -> None:
        part = await self.client.upload_part(
            Bucket=self.bucket,
            Key=self.key,
            PartNumber=self.part_number,
            UploadId=self.mpu["UploadId"],
            Body=chunk,
        )

        self.parts.append({"PartNumber": self.part_number, "ETag": part["ETag"]})
        self.part_number += 1

        self.uploaded_size += len(chunk) / 1024 / 1024
        print(self.uploaded_size)

    async def _finish_uploading(self) -> None:
        part_info = {"Parts": self.parts}
        await self.client.complete_multipart_upload(
            Bucket=self.bucket,
            Key=self.key,
            UploadId=self.mpu["UploadId"],
            MultipartUpload=part_info,
        )
        self.is_loading = False
```

Пример использования clients/fapi/s3.py:

```python
async def stream_file(self, bucket: str, path: str, file: str):
    async with self.session.create_client('s3', region_name='us-west-2',
                                          endpoint_url=self.endpoint_url,
                                          aws_secret_access_key=self.access_key,
                                          aws_access_key_id=self.key_id) as client:
        async with MultipartUploader(
                client=client, bucket=bucket, key=path
        ) as uploader:
            with open(file, 'rb') as fd:
                buf = fd.read(10 * 1024 * 1024)  # 10 MB
                while buf:
                    await uploader.upload_part(buf)
                    buf = fd.read(10 * 1024 * 1024)  # 10 MB
```