录屏接入USS技术方案

一. 概述

1. 背景

1. SRE提出需求：需要将terminal的操作记录录成视频并保存下来，视频将被上传储存在USS，以便未来下载进行回放
2. PRD地址：https://confluence.shopee.io/display/IPG/v1.10
3. 项目仓库地址：https://git.garena.com/shopee/sz-devops/fe/kubernetes-bff
4. 申请链接：https://stc.shopee.io/approval-center/create-ticket/Process_17a1f2d447d_3:60:1b00d899-3c90-11ec-9fd2-b49691a2ae86
5. USS介绍文档：https://confluence.shopee.io/pages/viewpage.action?pageId=585142901

2. 目标

1. 提供视频文件上传储存和下载

二. 技术方案选型

技术选型

- USS 选择

  1. 对象存储：扁平结构，每一个文件就是一个Object，通过一个个bucket来管理这些object

     适用场景：存储多媒体文件 / 图片处理，如裁剪，旋转，格式转换，缩放

  2. 文件存储：后面补充

- SDK 选择

  由于USS团队并没有封装Node SDK，我们有两个选择：

  1. 使用Go SDK，通过OpenApi访问USS服务

     优点：sdk一步到位

     缺点：请求链路变长

  2. 将HTTP接口进行封装，通过BFF直接访问uss服务

     优点：拓展性强

     缺点：封装配置比较麻烦，很多细节需要处理，比如etag算法

     要在bff层操作es数据库

我们并不在乎文件具体被存在哪里，简单的对象存储即可

NodeJS处理流数据更方便，所以决定用NodeJS来对USS接口进行封装

三. 核心设计

1. 架构设计

1. 边采集边上传
2. 采集完全后分片上传

建议选2，一个是uss限制，需要一个原始文件etag，另一个是断点续传会方便一点

- 上传（串行或并行？merge的时间差？并行数量限制？）

流式读取文件后，通过设置highWaterMark的值，限制每个chunk的大小，进行上传，获取各个chunk的file id

websocket断开链接后，最后一个chunk上传给USS，merge file之后返回一个fid，写入es (platform-backend-termlog-v4)

上传失败会每隔一秒再次发起请求，三次失败则不保存文件

前端根据fid是否存在来决定replay按钮是否置灰

单个文件大小限制为8GB

分片上传大小限制为1MB

- 下载

根据fid发起请求下载文件，bff处理后返回一个流

- 删除

通常来讲，object的数量没有限制，只要bucket不设置过期时间，不需要手动删除

如果一定要删除，可以在获取terminal log的时候进行判断，删除es数据的同时删除object

2. ES设计

要求1: command table里的每一条命令需要提供replay功能

现有的es的index platform-backend-termlog-v4的数据结构如下：

```
export const TERMINAL_LOG_MAPPING = {
  mappings: {
    properties: {
      '@timestamp': {
        type: 'date'
      },
      application: {
        type: 'keyword'
      },
      container: {
        type: 'keyword'
      },
      department: {
        type: 'text'
      },
      email: {
        type: 'keyword'
      },
      group: {
        type: 'keyword'
      },
      name: {
        type: 'keyword'
      },
      nodeIP: {
        type: 'keyword'
      },
      nodeName: {
        type: 'keyword'
      },
      pod: {
        type: 'keyword'
      },
      podIP: {
        type: 'keyword'
      },
      project: {
        type: 'keyword'
      },
      requestCmd: {
        type: 'text'
      },
      requestMethod: {
        type: 'text'
      },
      requestPath: {
        type: 'text'
      },
      requestQuery: {
        type: 'text'
      }
    }
  }
}
```

补充一个file id和当次websocket的sessionId

要求2: replay table记录每一次websocket连接

新的es index：platform-backend-terrminal-replay

数据结构如下

```
export const TERMINAL_REPLAY_MAPPING = {
  mappings: {
    properties: {
      '@timestamp': {
        type: 'date'
      },
      fileId: {
        type: 'keyword'
      },
      sessionId: {
        type: 'keyword'
      },
      application: {
        type: 'keyword'
      },
      container: {
        type: 'keyword'
      },
      email: {
        type: 'keyword'
      },
      group: {
        type: 'keyword'
      },
      name: {
        type: 'keyword'
      },
      nodeIP: {
        type: 'keyword'
      },
      nodeName: {
        type: 'keyword'
      },
      podName: {
        type: 'keyword'
      },
      podIP: {
        type: 'keyword'
      },
      project: {
        type: 'keyword'
      }
    }
  }
}
```

3. 权限设计

要求：

- Tenant Admin及Platform Admin可查看本Application下所有操作记录
- Developer可以查看自己的操作记录

USS暂不支持分权限访问，所有持有appId和appSecret的用户都能访问所有文件

建议在前端和bff进行权限控制

4. Bucket设计

bucket存活时间：可选，不选为无限期

bucket大小限制：无

bucket划分：

1. 根据部门划分（金融or非金融）
2. 根据读写频率划分（不需要）

建议只使用一个bucket

四. 风险+疑惑+问题记录

mergeFile需要提供原始文件的Etag，是否说明不支持边读边上传？而是必须等到读完一整个文件后，分片再上传？(待证实)（已证实不支持）

如果文件过大，计算etag值可能会很耗时（分片计算or web worker计算完返回给主线程）

USS项目接口过于简单， 缺少list object，ifChuckExist等接口

appId 和 appSecret的存放位置

断点续传的处理

bucket是否能分层（已证实不能）