文件下载与删除

文件下载流程方案：

1. 用户登录到主页面后要下载文件，点击要下载文件，就会显示下载或删除的按钮，用户点击下载的按钮，即可进行文件下载。
2. 前端获取用户要下载的文件的inode并将其发送给后端。
3. 后端接收前端发送的inode并在数据库中找到对应的加密的文件密钥、文件基本信息和数据块，将其发送给前端。
4. 前端接受后端发送的文件块信息，用登陆时储存在前端的主密钥将加密的文件密钥解密，并用文件密钥将加密的文件信息和数据块解密。
5. 根据文件块的顺序将文件块整合成一个完整的文件。

文件下载删除界面实现：

用户点击下载文件，  


文件下载部分代码详解：

1. 用户登录到主页面后要下载文件，点击要下载文件，就会显示下载或删除的按钮，用户点击下载的按钮，即可进行文件下载。

前端下载删除按钮实现：

<el-button

size="mini"

@click="download(scope.$index, scope.row)">下载

</el-button>

<el-button

size="mini"

type="danger"

@click="deleteFile(scope.$index, scope.row)">删除

</el-button>

1. 前端获取用户要下载的文件的inode并将其发送给后端。

下载任务：

download(index, row) {

var userId = this.userId

download({

userId,

row

})

1. 后端接收前端发送的inode并在数据库中找到对应的加密的文件密钥、文件基本信息和数据块，将其发送给前端。

在前端创建连接时时触发：

@OnOpen

    public void openSession(Session session, @PathParam("userId") Long userId) {

        this.session = session;

        BlockDownloadWebsockets.put(userId, this);

        addOnlineCount();

        log.info("创建连接:" + "当前连接数" + onlineCount);

    }

当后端接受字符消息（inode）时响应，后端接收inode，并在数据库中找到全部的文件块，并将这些文件块返回给前端：

@OnMessage

    public void onMessage(String message, @PathParam("userId") String userId) {

        log.info("字符串消息" + message);

        // 前端传进来一个字符串信息

        FilePo filePo = JSONObject.parseObject(message, FilePo.class);

        Set<FileBlockPo> allBlock = null;

        if (filePo.getFileBlocks() == null) {

            allBlock = blockService.findFileAllBlock(filePo);

        } else {

            allBlock = filePo.getFileBlocks();

        }

        for (FileBlockPo fileBlockPo : allBlock) {

            try {

                FileBlockPo downloadBlock = blockService.downloadBlock(fileBlockPo);

                this.sendMessage(JSONObject.toJSONString(downloadBlock, false));

            } catch (Exception e) {

                e.printStackTrace();

            }

        }

    }

在数据库去找到对应的加密的文件密钥、文件基本信息和数据块：

 /\*\*

     \* 下载文件块数据

     \* @param fileBlockPo

     \* @return

     \* @throws Exception

     \*/

    FileBlockPo downloadBlock(FileBlockPo fileBlockPo) throws Exception;

//在Ceph系统中找到所有的文件块

@Override

    public FileBlockPo downloadBlock(FileBlockPo fileBlockPo) throws Exception {

        S3Object object = amazonS3.getObject(bucketName, fileBlockPo.getFingerprint());

        S3ObjectInputStream objectContent = object.getObjectContent();

        InputStream delegateStream = objectContent.getDelegateStream();

        int b;

        List<Byte> res = new ArrayList<>();

        while ((b=delegateStream.read())!=-1){

            res.add((byte) b);

        }

        byte[] bytes = new byte[res.size()];

        for (int i = 0; i < res.size(); i++) {

            bytes[i] = res.get(i);

        }

        fileBlockPo.setData(bytes);

        return super.downloadBlock(fileBlockPo);

    }

接受到字节流流时创建，下载文件时前端无需传输字符流，若接收到则前端传输错误：

@OnMessage

    public void onMessage(byte[] message) {

        if (this.blockPo == null) {

            throw new ApiException(5000, "服务端错误");

        }

将文件下载完成断开连接时触发：

    @OnClose

    public void closeSession() {

        subOnlineCount();

        log.info("关闭socket通信:" + "当前连接数" + onlineCount);

    }

发生错误时触发：

    /\*\*

     \* 发生错误时触发

     \*

     \* @param session

     \* @param throwable

     \*/

    @OnError

    public void sessionError(Session session, Throwable throwable) {

        log.info("发生异常事件");

    }

1. 前端接受后端发送的文件块信息，用登陆时储存在前端的主密钥将加密的文件密钥解密，并用文件密钥将加密的文件信息和数据块解密。

解密列表数据：

async function encryptlist(tdata) {

let clientRandomValue = stringtoUint8Array(localStorage.getItem('clientRandomValue'));

const masterKey = stringtoUint8Array(localStorage.getItem('masterKey'));

for (var i = 0; i < tdata.length; i++) {

if (tdata[i].type === "DIR") {

tdata[i].size = '-'

} else {

var encryptedfileKey = stringtoUint8Array(tdata[i].fileKey)

var fileKey = await dec(masterKey, clientRandomValue, encryptedfileKey)

var encryptedfilename = stringtoUint8Array(tdata[i].filename)

var encryptedmtime = stringtoUint8Array(tdata[i].mtime)

console.log(encryptedfilename)

console.log(encryptedmtime)

console.log(fileKey)

fileKey = new Uint8Array([166, 211, 186, 222, 50, 173, 124, 208, 231, 185, 89, 47, 99, 253, 157, 56]);

clientRandomValue = new Uint8Array([28, 99, 105, 140, 100, 252, 242, 31, 114, 250, 121, 220, 43, 185, 162, 151]);

urn tdata

}

解密文件名：

var filename = await dec(fileKey, clientRandomValue, encryptedfilename)

console.log("文件名：" + filename)

tdata[i].filename = uint8ArrayToString(filename)

// var mtime = await dec(fileKey, clientRandomValue, encryptedmtime)

// console.log("mtime：" + mtime)

// tdata[i].mtime=dateToString(uint8ArrayToString(mtime).toDate())

tdata[i].size = showfilesize(tdata[i].size)

}

1. 根据文件块的顺序将文件块整合成一个完整的文件。

async clickFolder(row) {

if (row.type === "DIR") {

this.tableData = await encryptlist(row.childrenFiles)

this.curInode = row.inode

this.breadlist.push({"name": row.filename, "inode": row.inode})

}

},

next(name) {

var newpath = localStorage.getItem('path') + name + '/'

this.path = newpath

this.$http.post(this.$HOST + 'v2/filelist', this.$qs.stringify({

sign: this.$sign,

username: localStorage.getItem('name'),

path: newpath

})).then(res => {

localStorage.setItem('path', newpath)

this.tableData = []

res.data.data.dir.forEach(item => {

if (item.size == '') {

var size = '-'

} else {

if (item.size < 1048576) {

var size = (item.size / 1024).toFixed(2) + 'KB'

} else if (item.size > 1048576 && item.size < 1073741824) {

var size = (item.size / 1024 / 1024).toFixed(2) + 'MB'

} else if (item.size > 1073741824) {

var size = (item.size / 1024 / 1024 / 1024).toFixed(2) + 'GB'

}

}

this.tableData.push({name: item.name, time: item.mtime, img: item.img, size: size})

})

res.data.data.file.forEach(item => {

if (item.size == '') {

var size = '-'

} else {

if (item.size < 1048576) {

var size = (item.size / 1024).toFixed(2) + 'KB'

} else if (item.size > 1048576 && item.size < 1073741824) {

var size = (item.size / 1024 / 1024).toFixed(2) + 'MB'

} else if (item.size > 1073741824) {

var size = (item.size / 1024 / 1024 / 1024).toFixed(2) + 'GB'

}

}

this.tableData.push({name: item.name, time: item.mtime, img: item.img, size: size})

})

})

文件删除流程方案：

1. 用户登录到主页面后要删除文件，点击要删除的文件，就会显示下载或删除的按钮，用户点击删除的按钮，即可进行文件删除。
2. 前端获取要删除的文件的inode并将其发送给后端。
3. 后端接收前端发送的inode，并在数据库中找到改inode对应的文件信息并将其从数据库中删除，同时也要将Ceph系统储存的文件块一并找到并删除。
4. 后端删除成功后，前端将已经删除的文件从文件列表中删除。

文件下载删除界面实现：



文件删除部分代码详解：

1.用户登录到主页面后要删除文件，点击要删除的文件，就会显示下载或删除的按钮，用户点击删除的按钮，即可进行文件删除。前端按钮：

<el-button

size="mini"

@click="download(scope.$index, scope.row)">下载

</el-button>

<el-button

size="mini"

type="danger"

@click="deleteFile(scope.$index, scope.row)">删除

</el-button>

2.前端获取要删除的文件的inode，发送请求给后端，请求成功后，并将其发送给后端。

async clickFolder(row) {

if (row.type === "DIR") {

this.tableData = await encryptlist(row.childrenFiles)

this.breadlist.push({"name": row.filename, "inode": row.inode, "parent\_inode": this.curInode})

this.curInode = row.inode

}

},

// 删除文件

deleteFile(index, row) {

alert(index)

let \_this = this

request.delete("/files/" + this.userId + "/" + row.inode).then(function (res) {

if (res.code === 2000) {

alert("请求后端成功" + JSON.stringify(res))

\_this.tableData.splice(index, 1)

}

})

},

3.后端接收前端发送的inode，并在数据库中找到改inode对应的文件信息并将其从数据库中删除，同时也要将Ceph系统储存的文件块一并找到并删除。

    /\*\*

     \* 删除指定用户的指定文件

     \*

     \* @param inode  文件inode

     \* @param userPo 用户

     \*/

    void deleteByInodeAndUser(Long inode, UserPo userPo);

FilePo findByInodeAndUser(Long inode, UserPo userPo);

    /\*\*

     \* 删除文件

     \* @param inode 文件inode

     \* @param userId 用户id

     \*/

    void deleteFile(Long inode,Long userId);

4.后端删除成功后，前端将已经删除的文件从文件列表中删除。

//文件夹展示到当前目录

var time = dateToString(Mtime)

this.tableData.push({filename: name, mtime: time, size: '-', type: "DIR"})

async back() {

let b = this.fromData

for (var i = 0; i < this.breadlist.length - 1; i++) {

for (var j = 0; j < b.length; j++) {

if (this.breadlist[i].inode === b[j].inode) {

b = b[j].childrenFiles

}

}

}

// 删除文件

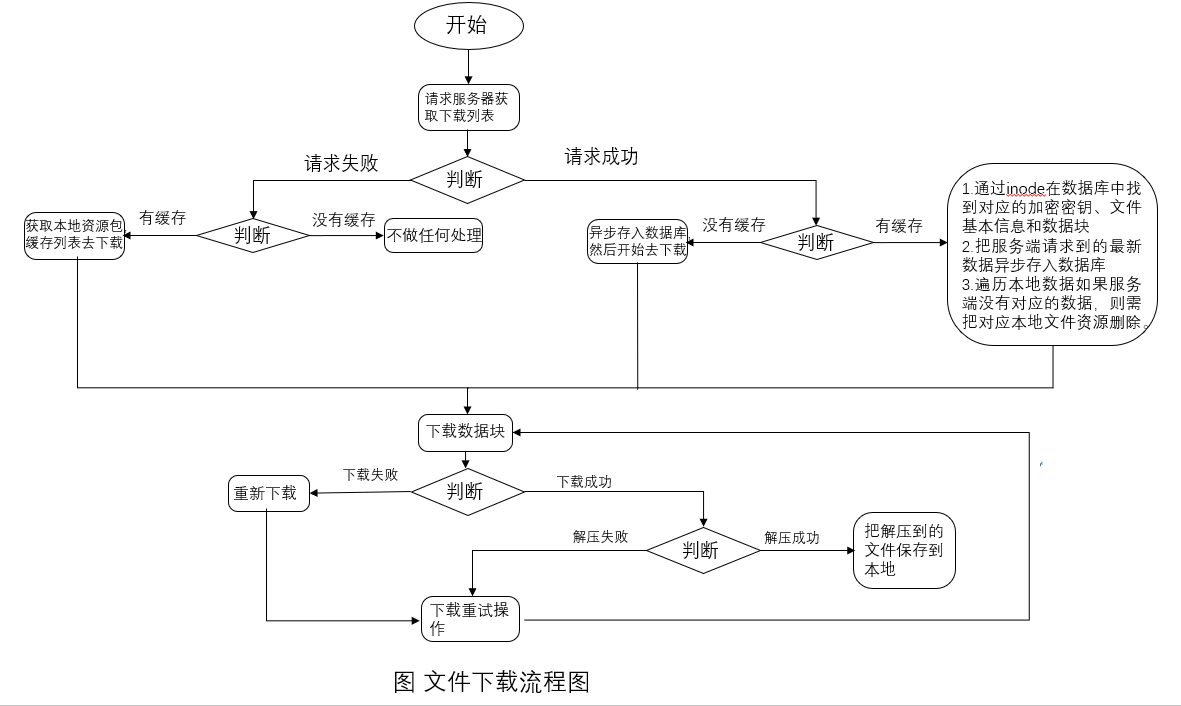
deleteFile(index, row) {

alert(index)

let \_this = this

request.delete("/files/" + this.userId + "/" + row.inode).then(function (res)

附文件下载流程图：



若需源码，请访问https://github.com/darKmili/cloud